Fraunhofer Institut
Software- und
Systemtechnik

Modellierung und dynamische Adaption klinischer Pfade auf Basis semantischer Prozessfragmente

Dissertation

Claudia Reuter
Dortmund, 18. Mai 2011

Kurzfassung

In Folge des hohen Kosten- und Qualitätsdrucks durch Politik, Krankenversicherungen und Patientenverbände, müssen Krankenhäuser ihre medizinischen und administrativen Prozesse vereinheitlichen und verschlanken, um wettbewerbsfähig zu bleiben. So genannte »Klinische Pfade« werden als Prozessstandards in Kliniken eingeführt; sie sollen dafür sorgen, dass Behandlungsabläufe besser planbar, kalkulierbar, dokumentierbar werden und sich qualitativ an den Erkenntnissen der evidenzbasierten Medizin orientieren. Zur Unterstützung der Einhaltung klinischer Pfade kommen zunehmend IT-Systeme, nämlich Workflow Management Systeme (WfMS) zum Einsatz.

Bisher wurden WfMS vornehmlich zur Steuerung industrieller Fertigungsprozesse genutzt. Die Natur solcher Prozesse unterscheidet sich von klinischen Prozessen in einem wesentlichen Punkt: Während in der Industrie das zu fertigende Produkt im Vordergrund aller Arbeiten steht, geht es im Gesundheitswesen um den Patienten und die Patientin, also Individuen, mit individuellen Bedürfnissen und Ansprüchen. Die Kernanforderung, die WfMS zur Unterstützung klinischer Prozesse erfüllen müssen, besteht also im geeigneten Umgang mit dem Spannungsverhältnis zwischen Prozessstandardisierung im Sinne von Planbarkeit, Kostenkalkulation und Qualität einerseits und Prozessflexibilisierung zur adäquaten Bedrücksichtigung fallbezogener Besonderheiten andererseits. Aktuelle Ansätze zum Umgang mit Varianz vom Standardprozess sind mit hohem Aufwand verbunden, führen mehrheitlich zu komplexen und unübersichtlichen Prozessdefinitionen und erfordern ein fundiertes technisches Know-How. Alle drei Aspekte sind Ausschlusskriterien für den flächendeckenden Einsatz von WfMS im klinischen Bereich.

Das hier erarbeitete Lösungskonzept basiert auf der Erkenntnis, dass fachliche Anwender Prozessabläufe nur dann dynamisch an die individuellen Besonderheiten eines Behandlungsfalles anpassen können, wenn sie zuvor selbst in der Lage waren, die ausführbare Prozessbeschreibung zu erstellen. Um die Voraussetzungen dafür zu schaffen, wird ein formales Domänenmodell spezifiziert, das der Verstetigung des Prozesswissens einer Einrichtung dient und aus dem sich sowohl klinische Pfade als Prozessstandards ableiten als auch flexibel Änderungen vornehmen lassen. Beim Domänenmodell handelt es sich um eine besondere und formale Variante des Feature-Modells, das sämtliche technische Anwendungskomponenten als Prozessfragmente in einen semantischen Kontext stellt, innerhalb dessen das Fragment ausgeführt werden kann; auf diese Weise werden aus unabhängig voneinander definierten und entwickelten Prozessfragmenten semantische Prozessfragmente (SPF). Die Erstellung und Anpassung von Prozessen erfolgt durch regelbasierte Komposition dieser Fragmente; die Regelmenge wird durch das Domänenmodell vorgegeben und ist als Graphgrammatik formalisiert. Fachliche Anwender können sich nun innerhalb des durch die Graphgrammatik aufgespannten Möglichkeitsraums bewegen, um technisch valide und ausführbare Prozessdefinitionen zu erzeugen. Die resultierenden Prozesse sind unabhängig von speziellen WfMS und Ausführungsumgebungen; das Lösungskonzept spezifiziert formale, systemübergreifende Regeln zur Transformation der Modelle in Prozessdefinitionen, die tatsächlich von einem WfMS ausgeführt werden können.

Zu den häufigen Ursachen für Abweichungen von klinischen Pfaden zählen Vorerkrankungen oder Komplikationen. Vorerkrankungen wie Diabetes mellitus oder Hypertonie müssen unabhängig von der aktuellen Einweisungsdiagnose und damit auch in unterschiedlichen klinischen

Pfaden berücksichtigt werden. Dasselbe gilt für eine Vielzahl von Komplikationen, wie z.B. Infektionen, die darüber hinaus zu unterschiedlichen Zeitpunkten während der Behandlung auftreten können. Das Lösungskonzept ist flexibel genug, um diese Besonderheiten sowohl während der Definition eines klinischen Pfades als auch nach seiner Instanziierung adäquat zu berücksichtigen. Es stellt jedoch auch die Mittel zur Verfügung, um für bestimmte Abweichungsursachen eigene Domänenmodelle zu kreieren; diese können genutzt werden, um Prozessvarianten als »Best practices« zur Modifikation klinischer Pfade in Folge einer speziellen Varianz anzulegen. In diesem Fall entsprechen die im Domänenmodell referenzierten semantischen Prozessfragmente also Änderungsoperationen auf den Prozessdefinitionen der Pfade. Die Anpassung eines klinischen Pfades als Reaktion auf das Feststellen der Vorerkrankung Diabetes melltitus reduziert sich für die Anwender anschließend auf die Auswahl der entsprechenden Variante. Analog zu den klinischen Pfaden selbst, können jedoch auch Prozessvarianten zuvor noch flexibel an individuelle Bedürfnisse der Patientinnen und Patienten angepasst werden.

Vorwort

Die vorliegende Arbeit enstand während meiner Tätigkeit als wissenschaftliche Mitarbeiterin am Fraunhofer Institut für Software- und Systemtechnik (ISST). Die Motivation für die Bearbeitung der in dieser Dissertation adressierten Problemstellung entwickelte sich bereits während der Betrachtung klinischer Pfade im Rahmen meiner Diplomarbeit; der Fokus hat sich während der Bearbeitung des Fraunhofer geförderten Projektes SPOT (Servicebasierte und Prozessorientierte Orchestrierungstechnologie) jedoch noch erweitert.

An dieser Stelle möchte ich mich bei allen Personen bedanken, die mich während der Arbeit unterstützt haben. Zuerst danke ich Prof. Dr. Jakob Rehof, der die universitäre Betreuung der Dissertation übernommen hat und die Entwicklungen mit großem Interesse begleitete. Prof. Dr. Peter Dadam danke ich für das Engagement und die Zeit, die er als zweiter Gutachter in die Dissertation investierte. Ich danke Dr. Wolfgang Deiters für seinen bedingungslosen Einsatz für die Arbeit, die Denkanstöße und nützlichen Hinweise beim Korrekturlesen. Besonderer Dank gebührt Jan Neuhaus, der von Beginn an als Rückhalt und Motivator unschätzbar für das Entstehen der Arbeit war.

Mein Dank gilt auch Dr. Andreas Beckers (Knappschaft-Bahn-See), der mir bereitwillig Zugang zu den klinischen Pfaden am Krankenhaus Bottrop gewährte und durch den ich auch Einblicke in das Pfadmodul des dort eingesetzten Krankenhausinformationssystems erhielt.

Ich danke Reza Eslami, der schon in einer sehr frühen Phase mit dem Korrekturlesen begonnen hat und von dem ich wertvolle Überarbeitungshinweise erhalten habe. Darüber hinaus möchte ich Manfred Wojciechowski, Boris Düdder und Thomas Königsmann danken, auf die ich mich als Sachverständige bezüglich aller organisatorischen Fragen zum Dissertationsablauf stets verlassen konnte.

Zu guter Letzt danke ich Matthias Krügl, der für alle Fragen, Probleme und Ideen in Zusammenhang mit der Arbeit immer ein offenes Ohr hatte.

Inhalt

1 Einleitung 9
1.1 Problemstellung 10
1.2 Vision und Zielsetzung der Dissertation 12
1.3 Aufbau der Arbeit 15

2 Anforderungsanalyse 17
2.1 Klinische Pfade 17
2.1.1 Hintergrund 17
2.1.2 Begriffsklärung und Abgrenzung 19
2.1.3 Kriterien zur Charakterisierung klinischer Pfade 21
2.1.4 Technische Umsetzung klinischer Pfade 24
2.1.5 Anforderungen an das Lösungskonzept 34
2.2 Varianz von klinischen Pfaden 36
2.2.1 Varianz am Beispiel der stationären Behandlung von Wirbelsäulenerkrankungen 37
2.2.2 Begriffsklärung und Definition 43
2.2.3 Anforderungen an das Lösungskonzept 46
2.3 Zusammenfassung der Anforderungen 52

3 Verwandte Arbeiten 57
3.1 Ansätze zur Entwicklung und Ausführung klinischer Pfade 57
3.1.1 Klinische Pfade bei der deutschen Rentenversicherung Knappschaft-Bahn-See 57
3.1.2 Klinische Pfade am ev. Krankenhaus Königin Herzberge 59
3.1.3 Theorie der kompositionalen Systemmodellierung 60
3.2 Ansätze zur Unterstützung manueller Abweichungsmaßnahmen 61
3.2.1 Abstraktion von Prozessdetails 62
3.2.2 Late Binding von Subprozessen 62
3.2.3 Late Modeling von Prozessteilen 65
3.2.4 Deklarative Prozessmodellierung 68
3.2.5 Nutzung von Änderungsprimitiven 70
3.2.6 Nutzung komplexer Änderungsoperationen 72
3.3 Ansätze zur Unterstützung von Prozessvarianten 75
3.3.1 Flexibilität durch Realisierung von Workflow Patterns 75
3.3.2 Festlegung von Ausführungsprioritäten 78
3.3.3 Konfiguration von Prozessdefinitionen 81
3.3.4 Kontextorientierte Konfiguration von Prozessinstanzen 84
3.3.5 Ereignisgesteuerte Umsetzung von Prozessvarianten 86
3.4 Zusammenfassung der Analyseergebnisse 89

4 Lösungskonzept 91
4.1 SPOT-Technologie: Lösung für ein anwendernahes Prozessmangement 91
4.1.1 DSL für integrierte Versorgungsprozesse 92
4.1.2 Kompositionale Prozessmodellierung 96
4.2 Überblick über das Lösungskonzept 99
4.3 Definitionsphase 102

4.3.1 Motivation für ein Domänenmodell als Basis klinischer Pfade 103
4.3.2 Feature-Modellierung zur Abbildung des prozessorientierten Domänenwissens 105
4.3.3 SPF-Typ-Graphen als Domänenmodelle für klinische Pfade 108
4.3.4 Formale Spezifikation von Prozessfragmenten 119
4.3.5 Formale Spezifikation von SPF-Typ-Graphen 125
4.3.6 Korrektheitskriterien für SPF-Typ-Graphen 129
4.3.7 Zusammenfassung der Ergebnisse 134
4.4 Konfigurationsphase 135
4.4.1 SPF-Graphen als Ergebnis der Konfiguration von SPF-Typ-Graphen 136
4.4.2 Graphgrammatiken auf Basis von SPF-Typ-Graphen 143
4.4.3 Berücksichtigung von Constraint-Relationen bei der Konfiguration 155
4.4.4 Verfahren bei der Konfiguration von SPF-Typ-Graphen 171
4.4.5 Zusammenfassung der Ergebnisse 177
4.5 Transformationsphase 178
4.5.1 Prozessdefinitionen als ADEPT2 WSM Nets und deren Korrektheitskriterien 180
4.5.2 Vorstellung der Transformations-relevanten Änderungsoperationen 184
4.5.3 Regeln für die Transformation von SPF-Graphen in WSM Nets 191
4.5.4 Ergänzung des Datenflusses auf Basis der Schnittstellenspezifikationen der Prozessfragmente 204
4.5.5 Transformation unvollständiger Konfigurationen von SPF-Typ-Graphen 210
4.5.6 Zusammenfassung der Ergebnisse 217
4.6 Rekonfigurationsphase 218
4.6.1 Rekonfiguration von SPF-Graphen zur Modellierungszeit 220
4.6.2 Rekonfiguration von SPF-Graphen zur Laufzeit 229
4.6.3 Identifikation von Knoten im SPF-Graphen zur Rekonfiguration 244
4.6.4 Zusammenfassung der Ergebnisse 249
4.7 Realisierung von Prozessvarianten 250
4.7.1 SPF-Änderungsoperationen zur Manipulation von SPF-Graphen 252
4.7.2 Formale Spezifikation von Prozessvarianten 264
4.7.3 Entwicklung konfigurierbarer Prozessvarianten auf Basis des Lösungskonzeptes 267
4.7.4 Zusammenfassung der Ergebnisse 271

5 Validierung des Lösungskonzepts an Hand einer Fallstudie 273
5.1 Definitionsphase 273
5.2 Konfigurationsphase 276
5.3 Transformationsphase 279
5.4 Rekonfigurationsphase 281
5.5 Prozessvariante »Diabetes mellitus« 284

6 Prototypische Umsetzung 291

7 Zusammenfassung & Diskussion 303

8 Ausblick 313
8.1 Validierung des Lösungskonzeptes im Kontext einer praktischen Fallstudie 313
8.2 Explizite Berücksichtigung weiterer Workflow-Aspekte bei der Konfiguration und Rekonfiguration von SPF-Typ-Graphen 314

8.3 Unterstützung des Prozesswechsels 314
8.4 Evolution von SPF-Typ-Graphen 315
8.5 Multi-Level Prozessmanagement 315

9 Referenzen 317

A Anhang und Verzeichnisse 341

1 Einleitung

Leistungserbringer und vor allem Krankenhäuser befinden sich unter zunehmendem Druck von Politik, Kostenträgern und Patientenverbänden, den ansteigenden Kosten im Gesundheitswesen Herr zu werden und gleichzeitig die Qualität der medizinischen Versorgung zu verbessern. Mangelnde Effizienz bei der Patientenbehandlung und die lückenhafte Abstimmung der Versorgungsprozesse werden als ein grundlegendes Problem genannt, das der Erreichung der Kosten- und Qualitätsziele entgegensteht [Bretschneider & Bohnet-Joschko 2007]. Schlecht abgestimmte und koordinierte Prozesse können tatsächlich negative Einflüsse auf alle beteiligten Personen und Institutionen haben. Stehen z. B. zum Zeitpunkt der stationären Aufnahme die Röntgenbilder des Patienten nicht bereit, so müssen radiologische Untersuchungen im Krankenhaus wiederholt werden; dies bedeutet zusätzliche Arbeit für das medizinische Personal, höhere finanzielle Aufwände für das Krankenhaus und eine unnötige Belastung für Patientinnen und Patienten. Gerade an den Schnittstellen zwischen unterschiedlichen Institutionen, stationären Abteilungen und Professionen ergeben sich durch mangelhafte Koordination der Arbeitsprozesse die meisten Schwierigkeiten.

Die Einführung eines umfassenden Prozessmanagements erscheint in vielen Einrichtungen die Antwort zu sein auf die Frage, wie sich der Spagat zwischen Kostensenkung und Qualitätssteigerung dauerhaft bewerkstelligen lässt. Prozessmanagement bedeutet im ersten Schritt die Offenlegung der tatsächlich stattfindenden Arbeitsabläufe und im zweiten Schritt die Identifikation von Optimierungspotential. Erst daraufhin werden die Mittel festgelegt, die sicherstellen sollen, dass die beschlossenen Optimierungen im praktischen Alltag tatsächlich umgesetzt und gelebt werden. Beispielsweise können einrichtungsinterne Dokumentationsrichtlinien festgelegt werden, an die sich das medizinische Personal halten soll. Da die Dokumentation der durchgeführten Leistungen jedoch bereits IT-gestützt erfolgt, liegt es nahe, IT-Systeme auch zur Prozesssteuerung einzusetzen. In Krankenhäusern wird das Prozessmanagement eng mit der Entwicklung und dem Einsatz so genannter »klinischer Pfade« verknüpft. Klinische Pfade entsprechen Prozessbeschreibungen, die darstellen, wie ein Patient ausgehend von einer bestimmten Diagnose, einer Prozedur oder einem Symptombild in einer konkreten Einrichtung behandelt werden soll. Form und Inhalt der Prozessbeschreibungen können von Krankenhaus zu Krankenhaus variieren.

Von der Einführung und Realisierung klinischer Pfade in Krankenhäusern erhofft sich das Management eine ganze Reihe von Vorteilen [Hindle 2007]; dazu zählen:

- Effiziente Organisation der Behandlungsabläufe zur Senkung der Liegezeiten von Patienten
- Transparente Darstellung der Prozessabläufe sowie der erweiterten Prozessinformationen, wie z. B. Personen- und Materialkosten zur Erleichterung der Kostenkalkulation und -kontrolle
- Kontinuierliche und durchgängige Dokumentation der erbrachten Leistungen zur Gewährleistung eines reibungslosen Abrechnungsvorgangs mit den Kostenträgern
- Ausrichtung von Behandlungsabläufen an den Erkenntnissen der evidenzbasierten Medizin (medizinische Leitlinien) zum Nachweis der Behandlungsqualität

Obwohl einige Einrichtungen mit der Einführung klinischer Pfade bereits Erfolge erzielt haben [Schilling et al. 2006, Thiel et al. 2006], konnten sich im Gesundheitswesen das Prozessmanagement generell und das IT-gestützte Prozessmanagement im Besonderen noch nicht flächendeckend durchsetzen. Im folgenden Kapitel wird untersucht, was die Hindernisse auf dem

Weg zum erfolgreichen IT-gestützten Prozessmanagement sind und mit welchen Schwierigkeiten speziell im Gesundheitswesen und in Krankenhäusern gerechnet werden muss.

1.1 Problemstellung

Prozessmanagement beginnt mit der Beschreibung der gewünschten Geschäftsprozesse als fachliche Abläufe durch die Domänenexperten. Häufig handelt es sich dabei um Dokumente in narrativer Form, seltener um informale Prozessmodelle, die unter Nutzung einer Modellierungssprache wie z. B. UML [ISO 2005] oder BPMN [OMG 2009] erstellt werden. In der Regel sind die so kreierten Geschäftsprozesse nicht mit IT-technischen Mitteln ausführbar, da für die Umsetzung durch Nutzung eines prozessorientierten IT-Systems, also eines Workflow Management Systems (WfMS), technische Informationen fehlen. Ausführbare Prozessdefinitionen müssen z. B. mit Informationen zu Anwendungskomponenten wie Web Services oder Benutzerschnittstellen angereichert werden, die für die tatsächliche Durchführung der Aktivität zuständig sind. Sollen die fachlichen Prozessbeschreibungen IT-gestützt realisiert werden, so müssen die informalen Beschreibungen händisch in (semi-)formale, ausführbare Prozessdefinitionen transformiert werden. Diese Aufgabe übernehmen in der Regel nicht die Domänenexperten oder die späteren Anwender, sondern IT-Experten, die jedoch meist wenig Kenntnisse über die fachlichen Abläufe besitzen und sich zunächst in das Anwendungsfeld einarbeiten müssen. Aber auch die Domänenexperten müssen sich zumindest teilweise mit den technisch geprägten Modellierungsumgebungen befassen, um die Modellierungsergebnisse der IT-Experten überhaupt bewerten zu können.

Dieses traditionelle Vorgehen ist nicht nur mit erheblichem Kommunikationsaufwand zwischen den Domänen- und IT-Experten sowie einem hohen Ressourceneinsatz verbunden, sondern auch sehr fehleranfällig. Der von den Domänenexperten entworfene Geschäftsprozess kann von technischen Mitarbeitern anders als ursprünglich intendiert interpretiert und umgesetzt werden. Genauso ist es möglich, dass der Geschäftsprozess unvollständig modelliert ist und wichtige Aspekte auslässt, die die IT-Experten nicht ohne Rückfragen ergänzen können. Die große Anzahl an Iterationen, die notwendig sind, um aus fachlichen Prozessbeschreibungen ausführbare Prozessdefinitionen zu generieren, ist Verständnisproblemen auf beiden Seiten geschuldet. Mangelhaftes Kommunikationsmanagement führt fast zwangsläufig zu mangelhaften Implementierungen der Geschäftsprozesse. Mit diesen Problemen werden letztlich die Anwender konfrontiert. Diese sind wiederum nicht in der Lage selbständig etwas an den Prozessdefinitionen zu verändern, um z. B. auf eine besondere Behandlungssituation zu reagieren, sondern können ihre Einwände lediglich an die Domänen- oder direkt an die IT-Experten weiterleiten. Letztlich führt dies zu einer eingeschränkten Nutzbarkeit des Systems, was wiederum negative Konsequenzen für die Akzeptanz hat. Der traditionelle Projektablauf für die Entwicklung von IT-Lösungen für das Prozessmanagement ist in Abbildung 1 dargestellt.

Abbildung 1 Prozessmanagement heute

Die Nummerierungen in der Graphik entsprechen den folgenden Projektphasen:

1. Erstellung der Prozessablaufbeschreibungen aus fachlicher Sicht durch die Domänenexperten
2. Kommunikation der Prozessabläufe durch die Domänenexperten an die IT-Experten
3. Überführung der Prozessablaufbeschreibungen in ausführbare Prozessdefinitionen durch die IT-Experten und Abstimmung der Ergebnisse mit den Domänenexperten
4. Endgültige Implementierung der Geschäftsprozesse unter Nutzung eines Workflow Management Systems und Freigabe der Anwendung
5. Test der Anwendung durch Instanziierung des Prozesses für bestimmte Anwendungsfälle sowie Kommunikation der Erfahrungen, Kritik und Verbesserungsvorschläge an die Domänenexperten

In der Gesundheitsdomäne und speziell im Krankenhaus treten die beim Prozessmanagement allgemein zu beobachtenden Probleme verstärkt zu Tage. Grund dafür ist zum einen, dass es sich bei der Gesundheitsdomäne um ein hoch profiliertes Anwendungsfeld handelt, das im Hinblick auf Fehlschläge bei der Einführung neuer IT-Systeme, wie z. B. WfMS, besonders sensibel reagiert. Das bedeutet, dass ein Fehlschlag die Reputation einer Systemkategorie oder auch der IT insgesamt dauerhaft schädigen kann [Yellowlees 2005]. Zum anderen handelt es sich bei den stark arbeitsteilig ausgeführten Behandlungsprozessen, die sowohl aus organisatorischen als auch pflegerischen und ärztlichen Aktivitäten bestehen, um extrem komplexe Abläufe, deren Abbildung mit Hilfe bekannter Prozessmodellierungsmethoden nur sehr schwer zu beherrschen ist. Das wahrscheinlich schwerwiegendste Problem besteht jedoch im Umgang mit dem hohen Maß an Flexibilität, die erforderlich ist, um trotz Standardisierungsbemühungen der Individualität jeder Behandlungssituation gerecht zu werden. Traditionelle

WfMS sind mehrheitlich auf die Unterstützung starrer Geschäftsprozesse ausgerichtet, die keine Anpassungen der Abläufe notwendig machen.

Der Einsatz solcher konventionellen Systeme würde bedeuten, dass klinische Pfade als »Happy Path« definiert werden, die nur dann eingehalten werden können, wenn die Behandlungssituation exakt den Rahmenbedingungen entspricht, unter denen der Behandlungsstandard festgelegt wurde. Alle Patienten, die nicht die Kriterien erfüllen, können dementsprechend auch nicht gemäß klinischem Pfad behandelt werden. Soll der Pfad dennoch genutzt werden, ist das medizinische Personal gezwungen, am System vorbeizuarbeiten; in diesem Fall würde die geplante und vom klinischen Pfad repräsentierte Behandlung jedoch nicht der tatsächlich durchgeführten Behandlung entsprechen. Beide Vorgehensweisen schränken die Nutzbarkeit und Akzeptanz des Systems zur Steuerung von Behandlungsprozessen stark ein und sind im Hinblick auf die Vollständigkeit der Behandlungsdokumentation nicht akzeptabel. Solange das Prozessmanagement auf Basis von WfMS umgesetzt werden soll, die flexible Abweichungen vom geplanten Prozessablauf nicht unterstützen, muss die Sinnhaftigkeit einer Investition in ein solches prozessorientiertes IT-System im Gesundheitswesen generell in Frage gestellt werden.

1.2 Vision und Zielsetzung der Dissertation

In Kooperation mit dem Fraunhofer IAO und dem Fraunhofer IML hat das Fraunhofer ISST ein Projekt initiiert und durchgeführt, das exakt die genannten Problemstellungen des Prozessmanagements adressiert und am Beispiel der Domänen Logistik und Gesundheitswesen Lösungen für diese Probleme entwickelt. Im SPOT-Projekt[1] (Service-basierte und Prozessorientierte OrchestrierungsTechnologie) wurde die These aufgestellt, dass Prozessmanagement nur dann effizient umgesetzt werden kann, wenn die Domänenexperten bzw. Fachanwender von vorneherein selbst in der Lage sind, ausführbare Prozessdefinitionen zu erstellen. Darüber hinaus wurde der Schluss gezogen, dass Anwender nur dann Prozesse an eine individuelle Situation anpassen können, wenn sie die Semantik der aktuell in Ausführung befindlichen Prozessinstanz und der ihnen zur Verfügung stehenden Änderungsoperationen verstehen. Das veränderte Vorgehen bei der Umsetzung von Prozessmanagement-Projekten wird in Abbildung 2 illustriert; dabei werden neue Prozesse oder Prozessänderungen von Fachanwendern per Knopfdruck dem IT-System übermittelt, das für die Prozessausführung zuständig ist.

[1] Siehe Projekt-Homepage unter http://www.spot.fraunhofer.de/

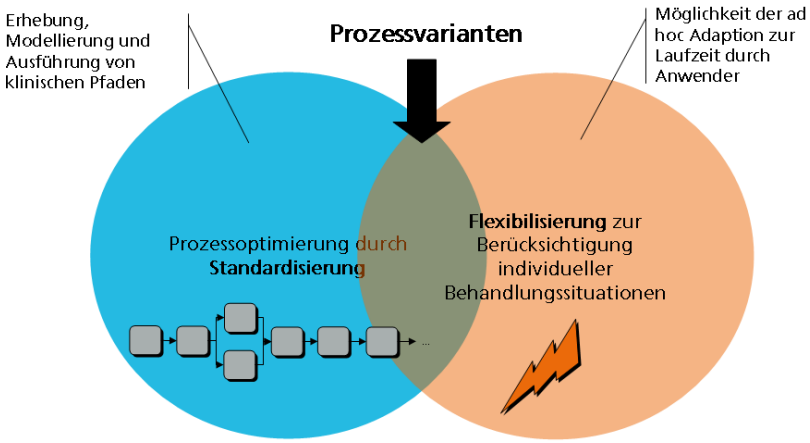
Abbildung 2 Prozessmanagement morgen

Die Entwicklung und dynamische Adaption ausführbarer Prozessdefinitionen durch Fachanwender setzt voraus, dass Modellierungs- und Ausführungsumgebungen für die Erstellung, Instanziierung und Bearbeitung von Prozessbeschreibungen an die individuellen Bedürfnisse der Domänenexperten adaptiert werden müssen. Einen Schwerpunkt der Arbeiten im SPOT-Projekt bildete daher die Identifikation von Lösungsbausteinen zur Schaffung solcher Modellierungs- und Ausführungsumgebungen. Da von den fachlichen Experten nicht erwartet werden kann, Anwendungskomponenten zu entwickeln oder Datenobjekte zu spezifizieren, wurde im SPOT-Projekt die Forderung nach Erstellung einer Bibliothek erhoben, in der das prozessorientierte Wissen einer Domäne oder eines Anwendungsbereiches abgelegt ist. Die Bausteine dieser Bibliothek können anschließend von Domänenexperten zur Modellierung neuer oder Anpassung vorhandener Prozessdefinitionen genutzt werden. Nicht zuletzt bedarf es jedoch auch der Möglichkeit, Prozesse, die sich bereits in Ausführung befinden, ad hoc zu modifizieren, sobald sich die Rahmenbedingungen ändern, unter denen der Prozessstandard entwickelt wurde.

Im Hinblick auf klinische Pfade existiert eine Menge von Gründen, die zu Abweichungen vom geplanten Behandlungsverlauf führen können. Dazu zählen z. B. Vorerkrankungen wie Diabetes mellitus oder Herzinsuffizienz, Komplikationen in Folge einer Prozedur, aber auch persönliche Wünsche der Patientinnen und Patienten oder ihrer Angehörigen. Diese Abweichungen werden in der Terminologie klinischer Pfade auch als »Varianz« bezeichnet. Mit der Einführung klinischer Pfade begeben sich Krankenhäuser zwangsläufig in das Spannungsfeld zwischen Standardisierung der Behandlungsabläufe und gleichzeitiger Flexibilisierung bei der Befolgung definierter Behandlungsstandards. Die Abweichung vom klinischen Pfad als Standardvorgehen ausgehend von einer spezifischen Ursache, mündet in der Erstellung einer Reihe von Prozessvarianten; das bedeutet, dass die operative Behandlung von Patienten mit Diabetes mellitus in Folge eines Bandscheibenvorfalls eine Variante des klinischen Pfades zur operativen Behandlung von Bandscheibenvorfällen darstellen kann. Bei der genauen Betrachtung von Varianz von klinischen Pfaden fällt auf, dass viele Ursachen für Abweichungen und ihre Konsequenzen für die weitere Gestaltung der Patientenbehandlung im Voraus bekannt sind und bereits bei der Erstellung der Prozessdefinition berücksichtigt werden können. In der nachfolgenden Abbildung, die die besondere Problematik bei der Realisierung klinischer Pfade

zum Ausdruck bringt, werden Prozessvarianten daher als Schnittmenge von Standardisierung und Flexibilisierung der Behandlungsabläufe angesiedelt.

Abbildung 3 Klinische Pfade im Spannungsfeld zwischen Standardisierung und Flexibilisierung



Die Machbarkeit der Vision hinter dem SPOT-Projekt, nämlich die Modellierung ausführbarer Prozessdefinitionen und die Möglichkeit zur dynamischen Anpassung von Prozessen durch die Fachanwender selbst, wird durch die Entwicklung einer prototypischen Realisierung (SPOT Healthcare Demonstrator [Reuter 2009]) demonstriert. Im Kontext dieses Prototypen wurde nicht nur eine domänenspezifische Sprache zur Modellierung von Behandlungsplänen für die integrierte, also sektorübergreifende gemeinschaftliche Versorgung von Patienten erstellt, sondern auch die Transformation der Behandlungspläne in ausführbare Prozessdefinitionen umgesetzt. Darüber hinaus besteht während der Ausführung des Prozesses auch die Möglichkeit, den Behandlungsplan ad hoc auf Basis einer vordefinierten Variante zu modifizieren. Sowohl die Selektion der Prozessvariante als auch die Transformation des Prozesses erfolgen jedoch beispielhaft für den ausgewählten Anwendungsfall. Im SPOT-Projekt wurde weder ein Modell zur Erfassung und Verwaltung des prozessorientierten Domänenwissens oder zur Ableitung von Prozessmodellen, noch ein generisches Konzept zur Modifikation von Prozessen oder zur Spezifikation von Prozessvarianten definiert. Diese Lücke soll im Rahmen dieser Dissertation geschlossen werden. Die Zielsetzung der Arbeit lässt sich demnach folgendermaßen zusammenfassen:

- Entwicklung eines formalen Modells und einer Vorgehensweise zur Abbildung von prozessorientiertem Domänenwissen
- Definition formaler Methoden zur Ableitung ausführbarer Prozessdefinitionen auf Basis des Domänenwissens
- Identifikation und Analyse notwendiger Flexibilisierungsanforderungen an klinische Pfade sowie Spezifikation entsprechender Lösungen
- Prototypische Realisierung von Teilen des Konzeptes

Im Hinblick auf das flexible bzw. adaptive Workflow Management wurde speziell in den letzten Jahren erheblicher Entwicklungsaufwand geleistet und beachtliche Ergebnisse erzielt. Diese Arbeit soll daher bereits existierende Lösungsansätze in diesem Umfeld in besonderem Maße berücksichtigen.

1.3 Aufbau der Arbeit

Die Arbeit beginnt in Kapitel 1 mit der Analyse der Anforderungen an ein Lösungskonzept zur prozessorientierten Umsetzung klinischer Pfade. Zunächst werden die Hintergründe zu klinischen Pfaden, ihre Eigenschaften und ihre technische Umsetzung betrachtet. Diesbezüglich werden WfMS als IT-Werkzeuge zur Modellierung und Ausführung der Pfade untersucht. Es wird dargestellt, wo die konkreten Potentiale dieser Technologie zur Unterstützung des Prozessmanagements im Krankenhaus liegen und was die Hindernisse sind, die ihrer flächendeckenden Nutzung im Wege stehen. Da mit der Etablierung klinischer Pfade der Versuch unternommen wird, organisatorische und medizinische Prozessabläufe in Krankenhäusern zu vereinheitlichen und zu optimieren, gerät das ärztliche, pflegerische und administrative Personal zwangsläufig in Situationen, in denen sich der Standard nicht mit den Anforderungen eines speziellen Behandlungsfalles deckt. Aus diesem Grund müssen sämtliche Bestrebungen in Richtung Standardisierung von Behandlungsprozessen immer mit Maßnahmen zur Flexibilisierung der Prozesse im Bedarfsfall flankiert werden. Dementsprechend wird das Phänomen der »Varianz«, also der begründeten Abweichung von klinischen Pfaden, detailliert untersucht und Anforderungen an die Unterstützung von Varianz durch WfMS abgeleitet. Die Ergebnisse der Gesamtanalyse werden in Kapitel 2.3 tabellarisch zusammengefasst.

Mit dem steigenden Bedarf nach Modellierung und Ausführung klinischer Pfade stellen sich erweiterte Anforderungen an die in Kliniken eingesetzten IT-Systeme, die dementsprechend weiterentwickelt wurden. Auch ist Prozessflexibilisierung in den letzten Jahren zum wichtigen Forschungsthema avanciert. Daher erfolgt in Kapitel 1 eine Präsentation existierender praktischer und wissenschaftlicher Lösungsansätze sowie die Diskussion ihrer Eignung zur Umsetzung flexibler, klinischer Pfade. Es werden sowohl Ansätze vorgestellt, die manuelle ad hoc Änderungen an laufenden Prozessinstanzen unterstützen, als auch solche, die eine Menge komplexer Änderungsoperationen am Standardprozess erlauben und damit Prozessvarianten realisieren.

Kern der Arbeit ist Kapitel 1, welches das in dieser Dissertation entwickelte Lösungskonzept zum Inhalt hat. Zu Beginn des Kapitels werden die Vorgängerarbeiten aus dem SPOT-Projekt dargestellt und gezeigt, dass noch großer Bedarf zur Konkretisierung und Formalisierung besteht. Das Konzept selbst gliedert sich in vier Phasen: In der Definitionsphase wird ein formales Modell zur Abbildung des domänenorientierten Prozesswissens erstellt. Dieses Modell bildet die Grundlage zur semantischen Annotation und Verknüpfung von Prozessfragmenten, die sich auf diese Weise zu »semantischen Prozessfragmenten« entwickeln. Ziel der Konfigurationsphase ist die Ableitung von konkreten Prozessen in deklarativer Form durch Konfiguration des Basismodells. Diese Prozessrepräsentationen korrespondieren noch nicht mit einer bestimmten Prozessmodellierungssprache und sind unabhängig von speziellen WfMS. Erst in der Transformationsphase werden sie in Prozessdefinitionen gemäß einer ausgewählten Modellierungssprache umgewandelt; in dieser Arbeit wird die Transformation am Beispiel von ADEPT2 WSM Nets durchgeführt. Durch die Konformitätsanforderung zum gemeinsamen Basismodell erhalten alle Prozessdefinitionen und ihre variablen Abläufe die notwendige Grundlage, auf der sich einfache Änderungen in der Rekonfigurationsphase aber auch komplexe Prozessvarianten umsetzen lassen.

Die Validierung des vorgestellten Lösungskonzeptes erfolgt in zwei Schritten: Zuerst wird in Kapitel 1 die Realisierung sämtlicher Phasen an Hand einer Fallstudie, nämlich der »stationären Behandlung von Wirbelsäulenerkrankungen« dargestellt. Anschließend wird in Kapitel 1 die prototypische Implementierung, die auf Basis des Konzeptes entwickelt wurde, beschrie-

ben. Zum Schluss werden die Ergebnisse der gesamten Dissertation zusammengefasst und diskutiert, ob die zuvor identifizierten Anforderungen durch das Lösungskonzept erfüllt werden; darüber hinaus werden Anregungen für weitere Forschungsarbeiten in diesem Umfeld gegeben.

2 Anforderungsanalyse

Bei klinischen Pfaden handelt es sich um das Instrument für Prozessmanagement im Krankenhaus. Neben einer Einführung in die Hintergründe und Grundlagen klinischer Pfade wird in diesem Kapitel mit Workflow Management Systemen (kurz: WfMS) eine Technologie zur Modellierung und Ausführung von Prozessen vorgestellt. Es wird diskutiert, warum sich diese Technologie in Krankenhäusern bisher nicht durchsetzen konnte und Anforderungen aufgestellt, die erfüllt werden müssen, um klinische Pfade zu unterstützen. Da die Standardisierung von Prozessen schnell an ihre Grenzen stößt, sobald es – wie im Fall klinischer Pfade – um die Behandlung von Menschen mit individuellen Bedürfnissen geht, besteht eine Kernanforderung in der Bereitstellung von Methoden, um flexibel auf Abweichungen vom Standard zu reagieren. Aufgrund ihrer Relevanz für die Entwicklung eines Lösungskonzepts zur prozessorientierten Umsetzung klinischer Pfade, wird im zweiten Teil des Kapitels speziell auf die so genannte »Varianz« von klinischen Pfaden eingegangen und Anforderungen an den geeigneten Umgang mit ihr abgeleitet. Das Kapitel endet mit einer zusammenfassenden Darstellung der Anforderungen an ein Lösungskonzept, das sowohl der Entwicklung klinischer Pfade als Standardprozesse als auch der adäquaten Behandlung von Varianz dient.

2.1 Klinische Pfade

Klinische Pfade definieren Standards für die Behandlung von Patientinnen und Patienten in Krankenhäusern. Obwohl ihre Einführung aus Qualitäts- und Kostengründen für Kliniken das Gebot der Stunde darstellt, fristen sie immer noch ein Schattendasein [Kleemann 2010]. Grund dafür sind erhebliche Hürden in Bezug auf Akzeptanz und Aufwand, die es zu überwinden gilt, sowie immer noch existierende Unsicherheiten hinsichtlich ihres konkreten Nutzens. Gerade der Einsatz von Informationstechnologie könnte dazu beitragen, Effizienz und Effektivität bei der Entwicklung, Ausführung und Analyse klinischer Pfade zu steigern; dennoch ist die WfM-Technologie noch nicht ausgereift genug, um die besonderen Anforderungen des Prozessmanagements in Krankenhäusern zu erfüllen. Warum das so ist, soll in diesem Kapitel analysiert werden.

2.1.1 Hintergrund

In Deutschland wurde mit der Verabschiedung des »Gesetzes zur Einführung des diagnoseorientierten Fallpauschalensystems für Krankenhäuser (Fallpauschalengesetz)« [BmG 2002] das Vergütungssystem grundlegend reformiert. Das Gesetz orientiert sich an dem 1983 in Australien und den USA eingeführten Modell der DRGs (Diagnosis Related Groups). DRGs bzw. Fallpauschalen dienen zur Klassifikation stationärer Behandlungen in klinisch definierte und nahezu aufwandshomogene Gruppen [Fischer 2001]. Als Definitionsgrundlage fungieren medizinische Diagnosen- und ergänzend Operationen- und Prozedurenschlüssel. Bei Bedarf werden im Einzelfall weitere Faktoren in die Berechnung mit einbezogen, z.B. Alter oder Geschlecht. Das Vorkommen unterschiedlicher Schweregrade wird durch die Unterscheidung nach Haupt- und Nebendiagnosen berücksichtigt. Mit der Abbildung des Leistungsspektrums von Krankenhäusern auf einen DRG-Katalog beabsichtigt der Gesetzgeber eine bundeseinheitliche Vergütung medizinischer Leistungen in Krankenhäusern und eine Abschaffung von Fehlanreizen für lange Verweildauern [BmG 2001].

Vor der Einführung von Fallpauschalen wurde die Dauer eines Krankenhausaufenthaltes eines Patienten mit einem tagesgleichen Pflegesatz vergütet, der mit den Kostenträgern ausgehandelt wurde und die entstandenen Ausgaben deckte. Auf diese Weise war es möglich, die Verweildauer von Patienten bei Bedarf flexibel zu verlängern, z.B. wenn die Unterbringung im Pflegeheim noch nicht geregelt war. Mit dem DRG-System jedoch wird die Krankenhausbehandlung nach ihrem ökonomischen Aufwand, d. h. ihrem Ressourcenverbrauch, finanziell gewichtet, was die bisher vorhandenen Spielräume im Hinblick auf Ausdehnung der Verweildauer deutlich begrenzt. Tatsächlich wird die Verweildauer von Patienten zu einem zunehmend kritischen Faktor, denn jeder Tag verursacht zusätzliche Kosten, die die Wirtschaftlichkeit des Krankenhauses schmälern. Es offenbart sich ein direkter Zusammenhang zwischen ärztlicher Verantwortung und ökonomischer Notwendigkeit. Das neue Gesetz ändert also nicht nur die Art und Weise, wie medizinische Leistungen in Krankenhäusern vergütet werden, sondern hat tatsächlich sehr weit reichende Auswirkungen auf deren Ablauf- und Aufbauorganisation: Die Kliniken sind gefordert, ihre bisherigen Prozesse auf den Prüfstand zu stellen und den Behandlungsablauf diagnoseorientiert und effizient zu restrukturieren, um wirtschaftlich arbeiten zu können.

Um der Gefahr vorzubeugen, dass die Qualität medizinischer Leistungen zu Gunsten der Gewinnmaximierung abnimmt, wird die Einführung von Fallpauschalen durch Regelungen zur Qualitätssicherung bzw. -steigerung flankiert. So legt der Gesetzgeber erweiterte Verpflichtungen zur Qualitätssicherung fest; dazu zählen z.B. die Erfüllung bundesweiter Mindestanforderungen an Struktur- und Ergebnisqualität sowie die Veröffentlichung strukturierter Qualitätsberichte. Auch die Tatsache, dass für Patienten, die aufgrund einer Komplikation in Zusammenhang mit der durchgeführten Leistung wieder in dasselbe Krankenhaus aufgenommen werden, keine erneute Fallpauschale angesetzt werden darf, erhöht die Qualitätsanreize [BmG 2001]. Die regelmäßige Qualitätsmessung ermöglicht Patienten außerdem zum ersten Mal an Hand objektiver Kriterien Krankenhäuser miteinander zu vergleichen und eine bewusste und selbst bestimmte Wahl zu treffen. Gleichzeitig verursacht sie einen Konkurrenzkampf zwischen den Häusern. Kliniken sind gezwungen, ihre Leistungsqualität einer kontinuierlichen Analyse zu unterziehen und sie zu verbessern, um dauerhaft wettbewerbsfähig zu bleiben.

Eine wichtige Voraussetzung zur Verbesserung der Behandlungsqualität bei gleichzeitiger Kostensenkung sehen Experten in der Reduktion der Variationen in der Patientenversorgung [Panella et al. 2003, Bevan & Cawley 2006]. Denn Variationen führen häufig nicht nur zu höheren Ausgaben für das Krankenhaus, sondern können auch negative Folgen für die Patienten haben. Chassin und Galvin [Chassin & Galvin 1998] publizierten zu diesem Thema ein Konsensuspapier (National Roundtable on Health Care Quality des Institute of Medicine der National Academy of Sciences der USA), in dem sie drei wesentliche Problemfelder im Hinblick auf die medizinische Versorgungsqualität in den USA herausstellen:

- Überversorgung von Patienten durch zu viel Diagnostik und zu lange Aufenthaltsdauern
- Fehlversorgung von Patienten durch Fehler in der Medikation und die Durchführung falscher Therapien
- Unterversorgung von Patienten durch mangelnde oder unregelmäßig durchgeführte Kontrollen oder Nicht-Behandlung (z.B. Bluthochdruck)

In Deutschland kommt der Sachverständigenrat für die Konzertierte Aktion im Gesundheitswesen zu demselben Ergebnis [Sachverständigenrat 2010]. McNeil [McNeil 2001] zieht den Schluss, dass diese Probleme einerseits durch die Unsicherheit der behandelnden Ärzte hinsichtlich Diagnostik und Therapie und zum anderen durch den ungesteuerten Einsatz neuer Technologien verursacht werden. Aus diesem Grund sind Instrumente erforderlich, die den

Leistungserbringern mehr Sicherheit geben, der Umsetzung von Qualitätsrichtlinien dienen und zum wirtschaftlichen Handeln motivieren. Außerdem ist es ohne Standardisierung der Behandlung nur unter sehr hohem Aufwand möglich, die Kostendeckung durch die DRG-Erlöse nachzuvollziehen.

Bei klinischen Pfaden handelt es sich um ein strategisches Instrument zur Sicherung bzw. Steigerung der Behandlungsqualität sowie zur Erleichterung der Kostenkalkulation und –kontrolle durch Standardisierung der Behandlungsabläufe. Demzufolge ist die Entwicklung und Umsetzung klinischer Pfade eine direkte Folge der Einführung des Fallpauschalen-Systems: »Wäre der Einführung der DRG in unseren Krankenhäusern in den USA nicht der Aufbau der Clinical Pathways gefolgt, wären die meisten heute nicht mehr existent« [Paeger 2001].

2.1.2 Begriffsklärung und Abgrenzung

Das erste Mal wird der Begriff »Clinical Pathway« in Zusammenhang mit der Steuerung der Patientenbehandlung 1985 von Zander und Bower am New England Medical Center (Boston, USA) genutzt. Inzwischen hat das Konzept klinischer Pfade international sowohl in der Literatur als auch in der Praxis Beachtung gefunden [Zander 2002, Hindle & Yazbeck 2005, Vanhaecht et al. 2006]. Eine eindeutige Definition klinischer Pfade fällt jedoch schwer, da viele unterschiedliche Begrifflichkeiten existieren, die teilweise identisch und teilweise divergent interpretiert werden. Beispiele für Begriffe sind »Behandlungspfade, Leitlinien, Behandlungsrichtlinien, Patientenpfade, Clinical Pathways, Critical Pathways, Care Pathway, Integrated Care Pathway, Standard Operating Procedures etc« [Roeder et al. 2003b]. In England finden sich ca. 30 unterschiedliche Begriffe allein für Behandlungs- und Pflegepfade, in den USA werden ca. 200 Begriffe verwendet [Wilson 1997]. Aktuell wird im deutschsprachigen Raum die Terminologie klinischer Pfad synonym zu »Clinical Pathway« oder »Critical Pathway« gebraucht [Holler et al. 2002]. Die folgende Tabelle gibt einen Überblick über einige Definitionen für klinische Pfade.

Tabelle 1 Überblick über Definitionen für klinische Pfade

Literaturreferenz	Definition
Bryan et al. 2002	»a map of the process involved in managing a common clinical condition or situation. It should detail what to do, when to do it, by whom the action should be untertaken and where the task should be performed«
Drumm & Achenbach 2005	»Ein integrierter klinischer Behandlungspfad ist die Beschreibung einer kompletten, interdisziplinär und/oder sektorenübergreifend erbrachten Behandlungsleistung für einen definierten Patiententyp. Der Behandlungspfad berücksichtigt Patientenanforderungen, den aktuellen Stand der medizinischen Erkenntnisse, die erforderliche Qualität der Leistungserbringung sowie Aspekte der Wirtschaftlichkeit. Er steuert den Leistungserstellungsprozess und unterstützt die Erfassung relevanter Daten zur Erhebung von organisatorischen, medizinischen und ökonomischen Abweichungen mit dem Ziel der kontinuierlichen Verbesserung«
European Pathway Association 2005	»Care pathways are a methodology for the mutual decision making and organization of care for a well-defined group of patients during a well-defined period«

Greiling et al. 2003	»Klinische Pfade sind abteilungs-, berufsgruppen-, und professionsübergreifend, medizinisch und ökonomisch abgestimmte Handlungsleitlinien für den gesamten Behandlungsablauf einer Gruppe homogener Behandlungsabläufe«
Hellmann 2002	»Ein klinischer Pfad ist ein netzartiger, berufsgruppenübergreifender Behandlungsablauf auf evidenzbasierter Grundlage (Leitlinien), der Patientenerwartung, Qualität und Wirtschaftlichkeit gleichermaßen berücksichtigt«
Müller et al. 2001	»Patientenpfade sind Standards für klinisch definierte Gruppen von Patienten auf der Basis von: Guidelines, Evidenzbasierter Medizin, Eigenerfahrung und klinikeigenen Standards«
Roeder et al. 2003a	»Ein klinischer Behandlungspfad ist der im Behandlungsteam selbst gefundene berufsgruppen- und institutionenübergreifende Konsens bezüglich der besten Durchführung der Krankenhaus-Gesamtbehandlung unter Wahrung festgelegter Behandlungsqualität und Berücksichtigung der notwendigen und verfügbaren Ressourcen sowie unter Festlegung der Aufgaben und der Durchführungs- und Ergebnisverantwortlichkeiten. Er steuert den Behandlungsprozess, ist gleichzeitig das behandlungsbegleitende Dokumentationsinstrument, und erlaubt die Kommentierung von Abweichungen von der Norm zum Zwecke fortgesetzter Evaluation und Verbesserung«
Thiemann 1996	Ziel eines Clinical Pathways ist es, aus einer interdisziplinären Perspektive heraus die Patientenerwartung zu erkennen, Ereignisse, die für die Verweildauer verantwortlich sind, zu entdecken und Methoden zu entwickeln, welche die Qualität und Kosteneffizienz in der Patientenbehandlung gleichermaßen berücksichtigen«
Vogel et al. 2002	»Ein Patientenpfad ist eine institutionsübergreifende Leitlinie, die den Behandlungsablauf berufsgruppenübergreifend von der Annahme bis zur Entlassung beschreibt, der für die Mehrzahl der Patienten mit der entsprechenden Diagnose zutreffend ist und der die für den Krankenhausaufenthalt anfallenden Leistungen und Ressourcen prozessbezogen erfasst«

Den hier aufgeführten Definitionen zu Folge, werden klinische Pfade für spezifische Behandlungssituationen mit einer definierten Anzahl an Akteuren entwickelt, berücksichtigen aktuelle Qualitätsstandards und haben für alle Beteiligten ein gewisses Maß an Verbindlichkeit. Dennoch sind begründete Abweichungen vom klinischen Pfad möglich. Unter diesem Blickwinkel lassen sich die Pfade auch von medizinischen Richtlinien und medizinischen Leitlinien abgrenzen, mit denen sie oftmals fälschlicherweise gleichgesetzt werden.

Bei medizinischen Richtlinien handelt es sich um »von einer rechtlich legitimierten Institution konsentierte, schriftlich fixierte und veröffentlichte Regelungen des Handelns oder Unterlassens, die für den Rechtsraum dieser Institution verbindlich sind und deren Nichtbeachtung definierte Sanktionen nach sich zieht« [Nüllen & Noppeney 2006]. Medizinische Richtlinien haben demzufolge eine sehr hohe Verbindlichkeit für Ärzte, die verpflichtet sind, die vorgeschriebenen Handlungen einzuhalten und die Richtlinien nicht nur als Entscheidungshilfe anzusehen. Als Beispiel sei an dieser Stelle das am 01. Dezember 1997 in Kraft getretene »Transplantationsgesetz der Bundesärztekammer mit den Richtlinien zur Feststellung des Hirntodes - Dritte Fortschreibung 1997 mit Ergänzung gemäß Transplantationsgesetz (TPG)« [Angstwurm et al. 1998] genannt. Klinische Pfade haben nicht denselben Verbindlichkeitsanspruch wie medizinische Richtlinien, sondern lassen begründete Abweichungen von dem beschriebenen Behandlungsstandard zu.

Medizinische »Leitlinien sind systematisch entwickelte Aussagen zur Unterstützung der Entscheidungsfindung von Ärzten, anderen im Gesundheitssystem tätigen Personen und Patienten. Das Ziel ist eine angemessene gesundheitsbezogene Versorgung in spezifischen klinischen Situationen« [DDCZ 2007]. Im Gegensatz zu klinischen Pfaden werden medizinische Leitlinien unabhängig von konkreten Gesundheitsinstitutionen entwickelt und repräsentieren ein fundiertes Erfahrungswissen (evidenzbasierte Medizin) zu einem gesundheitlichen Problem; sie geben Handlungsempfehlungen, die Ärzte und Patienten bei der Diagnose und der Therapie unterstützen, ohne sie in ihrer Entscheidungsfindung einzuschränken. Die Evidenzbasierung erfordert die periodische Überprüfung und Bearbeitung der Leitlinien, um die Anpassung an aktuelle Forschungsergebnisse zu gewährleisten. In Deutschland werden medizinische Leitlinien von mehreren Institutionen entwickelt und publiziert. Die wichtigsten Einrichtungen sind die Bundesärztekammer (BÄK), die Kassenärztliche Bundesvereinigung (KBV), die Arbeitsgemeinschaft der Wissenschaftlichen Medizinischen Fachgesellschaften (AWMF) und das ärztliche Zentrum für Qualität in der Medizin (ÄZQ).

Bei der Entwicklung klinischer Pfade spielen medizinische Richtlinien und medizinische Leitlinien eine wichtige Rolle, da sie das wissenschaftliche und medizinische Fundament des Pfades bilden. Klinische Pfade bedeuten jedoch mehr als evidenzbasierte Leitlinien oder institutionsinterne Richtlinien, da sie neben medizinischen, qualitätsorientierten Handlungsanweisungen auch ökonomische Aspekte umfassen.

2.1.3 Kriterien zur Charakterisierung klinischer Pfade

Aufgrund der Vielzahl an unterschiedlichen Begrifflichkeiten und Definitionsmöglichkeiten für klinische Pfade, wird sich diese Arbeit nicht an einer konkreten Definition orientieren, sondern Pfade an Hand einer Liste von Kriterien charakterisieren, die in Tabelle 2 zusammengefasst sind und die in diesem Kapitel näher erläutert werden.

Tabelle 2 Kriterien zur Charakterisierung klinischer Pfade

Kriterium	**Ausprägung**
Initiative	Management / Personal eines Krankenhauses bzw. eines Klinikverbandes / einer Klinikkette
Einsatzort	Entwicklung und Nutzung innerhalb eines Krankenhauses
Verantwortlichkeiten	Pfad- bzw. Case Manager, Mitwirkung und Zuarbeit durch ein multidisziplinäres Team bestehend aus allen, die an dem Behandlungsablauf beteiligt sind
Grundlage für die Pfadentwicklung	Ist-Abläufe im Krankenhaus; geplante Soll-Abläufe im Krankenhaus; Evidenzbasierte Medizin (medizinische Leitlinien); medizinische Richtlinien
Darstellungsform	Narrativ, tabellarisch, modellbasiert
Inhalte	Organisatorische, ärztliche und pflegerische Aktivitäten und Entscheidungen; Behandlungsziele; Ein- und Ausschlusskriterien; Varianten; Datenobjekte; Ressourcen; Verantwortlichkeiten; Zeitangaben; Kosten
Zeitraum	Überschaubarer, abgeschlossener Zeitraum
Ziele	Steuerung der standardisierten Behandlung von Patienten; Verbesserung der interdisziplinären Kommunikation und Koope-

	ration; Sicherstellung von Qualitätsstandards; Optimierung der Ressourcenplanung und des Ressourcenverbrauchs; Kostenkontrolle
Zielgruppe	Homogene Gruppe von Patienten (Ein- und Ausschlusskriterien orientieren sich z.B. an Diagnosen, Prozeduren, Schweregrad)
Verbindlichkeit	Möglichkeit zur begründeten Abweichung

Die Entwicklung klinischer Pfade wird initiiert durch Vertreter eines Krankenhauses oder einer Gruppe von Kliniken, wie z.B. Management, ärztliche oder pflegerische Leitung. Der klinische Pfad wird für eine konkrete Institution erstellt und an die jeweiligen Erfordernisse und geltenden Rahmenbedingungen vor Ort angepasst. Charakteristisch für klinische Pfade ist die Zusammenarbeit bei der Patientenbehandlung in einem multidisziplinären und eventuell auch stationsübergreifenden Team. Dementsprechend sind an der Erstellung des klinischen Pfades auch alle betroffenen Berufsgruppen beteiligt. Mit der Aufgabe des Managements der Pfadentwicklung, der Organisation von Schulungen für Mitarbeiterinnen und Mitarbeiter sowie der konkreten Umsetzung des Pfades wird meist ein Pfad- bzw. Case Manager betraut.

Die Erstellung des klinischen Pfades erfolgt auf Basis der Erfassung des Ist-Zustandes des Behandlungsablaufs innerhalb des Krankenhauses. Dennoch verändern sich die Prozesse häufig bereits mit ihrer Modellierung in Form klinischer Pfade, da z.B. Verantwortlichkeiten für die Durchführung von Aktivitäten erstmals offiziell festgelegt werden. Auch spiegelt das Modell des klinischen Pfades nicht mehr den Ist-Zustand wieder, sondern entspricht bereits einem Soll-Ablauf, wenn z.B. im Rahmen der Prozesserhebung organisatorische Mängel oder unnötige medizinische Tätigkeiten identifiziert und ausgeräumt werden. Neben den Gegebenheiten vor Ort spielen auch die aktuelle wissenschaftliche Evidenz in Form medizinischer Leitlinien oder medizinische Richtlinien eine zentrale Rolle bei der Erstellung klinischer Pfade. Durch eine Verknüpfung mit Inhalten aus Leitlinien lässt sich die Einhaltung der geforderten Behandlungsqualität an Hand des Pfades nachweisen.

Darstellungsformen für klinische Pfade sind sehr unterschiedlich [Meiler 2005]. Für die initiale Beschreibung werden meist narrative oder tabellarische Darstellungen gewählt. Sobald die Ausführung klinischer Pfade jedoch durch IT-Systeme unterstützt werden soll, überwiegen modellbasierte Ansätze, wie z.B. Ablaufdiagramme und klinische Algorithmen. Die Inhalte sind bei allen Darstellungsformen weitgehend identisch. Generell umfassen klinische Pfade die Aktivitäten, die im Zuge der Behandlung durchgeführt werden sollen. Des Weiteren definieren sie die Reihenfolge, in der Aktivitäten ausgeführt werden, deren zeitliche Dauer, Verantwortlichkeiten und anfallende Kosten. Darüber hinaus sind Angaben zu den benötigten personellen und materiellen Ressourcen sowie zu Dokumenten, Informationen, Aufträgen und Leistungen üblich. Da die Verweildauer das grundlegende Kriterium für die Prozessplanung und Kostenrechnung ist, orientiert sich die Anordnung der Aktivitäten häufig an Behandlungstagen. Alle Aktivitäten des Pfades spielen sich in einem überschaubaren, abgeschlossenen Zeitraum ab. Pro Aktivität oder Behandlungstag können auch die zu erreichenden Ziele angegeben werden, an denen sich das Personal orientieren soll. Die Festlegung der Ziele richtet sich sowohl nach den Pfad spezifischen Behandlungszielen als auch nach den übergeordneten Geschäftszielen des jeweiligen Krankenhauses. Sie beziehen sich z.B. auf den Gesundheitszustand des Patienten wie Fieberfreiheit oder Schmerzfreiheit, auf die Patientenzufriedenheit, auf die Verweildauer oder die Behandlungsqualität wie Vermeidung von Komplikationen. Ein- und Ausschlusskriterien werden für jeden Pfad und bei Bedarf auch für Aktivitäten definiert; erfüllt ein Patient die Kriterien, zählt er zu der Fallgruppe, die gemäß klinischem Pfad behandelt werden darf bzw. für die eine spezifische Aktivität ausgeführt werden soll. Innerhalb eines klinischen

Pfades kann es auch mehrere Alternativen geben, die in Abhängigkeit der individuellen Behandlungssituation des Patienten gewählt werden; z.B. Implantation eines neuen oder Wechsel eines existierenden Herzschrittmachers.

Ziel bei der Entwicklung klinischer Pfade ist die Erstellung eines Standards für die Patientenbehandlung. Die Standardisierung erleichtert die Abstimmung zwischen den behandelnden Gruppen, die Integration von Qualitätsrichtlinien sowie die Kalkulation und Kontrolle der Kosten. Indem klinische Pfade Prozesse in Krankenhäusern transparent machen, dienen sie auch als Grundlage zur Optimierung der Ressourcenplanung und des Ressourcenverbrauchs.

Die Spezifikation einer homogenen Gruppe von Patienten an Hand von definierten Kriterien ist notwendig, um entscheiden zu können, ob der Patient gemäß klinischem Pfad behandelt werden kann. Die Zuordnung zu einer Fallgruppe erleichtert jedoch auch die Abrechnung der Behandlung nach Fallpauschalen. Grundsätzlich kann zwischen folgenden Gruppierungskriterien unterschieden werden [Keun & Prott 2008]:
- Diagnosebezogene Fallgruppenbildung
- Therapiebezogene Fallgruppenbildung
- Kombination aus diagnose- und therapiebezogener Fallgruppenbildung

Die diagnosebezogene Fallgruppenbildung geht von der Annahme aus, dass der Behandlungsbedarf des Patienten ausgehend von der Diagnose hinreichend beschrieben werden kann. Wesentliches Charakteristikum der Fallgruppe ist daher die Hauptdiagnose als Ursache des stationären Aufenthalts. Um das Krankheitsbild der Fallgruppe angemessen zu repräsentieren, werden bei Bedarf zusätzlich Nebendiagnosen angegeben. Die standardisierte Kodierung von Diagnosen erfolgt mit Hilfe des ICD-10 Katalogs (International Classification of Diagnoses and Related Health Problems - Internationale statistische Klassifikation der Krankheiten und verwandter Gesundheitsprobleme)[2]. Im Vordergrund der therapiebezogenen Fallgruppenbildung steht der durchzuführende medizinische Eingriff. Für dessen Verschlüsselung steht der OPS Katalog (Operationen- und Prozedurenschlüssel) zur Verfügung[3]. Eine diagnose- und therapiebezogene Fallgruppenbildung erfolgt, wenn die Diagnose allein nicht zur Klassifizierung des Patienten ausreicht, sondern sich die Eingruppierung am Leistungsgeschehen des Krankenhauses orientieren soll. In diesem Fall werden also das Symptombild des Patienten, sowie Diagnosen und Therapieverfahren gemeinsam betrachtet. Diagnosis Related Groups (DRGs) werden durch solche Kombinationen aus Diagnose und Therapie beschrieben. Weitere Einflussfaktoren im Hinblick auf die Fallgruppierung und die Bestimmung der Höhe der Fallkosten sind z.B. Alter und Geschlecht des Patienten oder auch die Schwere des Falles.

Das Klassifikationssystem nach Patient Management Categories (PMC) stellt eine Alternative zu DRG dar, bei dem nicht nach Haupt- und Nebendiagnose unterschieden wird, sondern alle

[2] Vgl. Deutsches Institut für medizinische Dokumentation und Information, DIMDI (Hrsg. im Auftrag des Bundesministeriums für Gesundheit unter Beteiligung der Arbeitsgruppe ICD-10 des Kuratoriums für Fragen der Klassifikation im Gesundheitswesen (KKG)): ICD-10-GM Version 2009 Systematisches Verzeichnis, Internationale statistische Klassifikation der Krankheiten und verwandter Gesundheitsprobleme, 10. Revision - German Modification - , Stand: 24.09.2008, www.dimdi.de

[3] Vgl. Deutsches Institut für Medizinische Dokumentation und Information, DIMDI (Hrsg. im Auftrag des Bundesministeriums für Gesundheit unter Beteiligung der Arbeitsgruppe OPS des Kuratoriums für Fragen der Klassifikation im Gesundheitswesen (KKG): OPS Version 2009 Systematisches Verzeichnis, Operationen- und Prozedurenschlüssel, Internationale Klassifikation der Prozeduren in der Medizin (OPS), Stand: 20.10.2008, www.dimdi.de

Diagnosen ohne Rangordnung berücksichtigt werden. Mehrere Diagnosen wurden zu insgesamt 54 Krankheitsmodulen zusammengefasst, denen Patienten je nach Schweregrad ihrer Erkrankung zugeordnet werden können. Schweregradkonzepte beziehen sich z.B. auf Multimorbidität, die Schwere der einzelnen Krankheit, die Schwere der Erkrankung insgesamt oder Zusatzprobleme bei der Pflege [Fischer 1997].

Bei der Zuordnung von DRGs zu klinischen Pfaden ist an Hand der gewählten Kodierung sofort ersichtlich, wie hoch die Vergütung der Patientenversorgung sein wird. Werden für die Aktivitäten innerhalb des klinischen Pfades Kostenstellen angegeben, ist auch eine Verrechnung der Kosten mit der Fallpauschale möglich. Bei einer Berechnung nach PMCs, gibt es zu jeder PMC einen Pfad, dessen Aktivitäten mit Kostengewichten versehen sind. Wurden für einen Patienten mehrere PMCs ausgewählt, so werden die Pfadaktivitäten verglichen und jeweils das höchste Kostengewicht angesetzt. Indem klinische Pfade Behandlungsabläufe und Kostenzusammenhänge im Krankenhaus transparent machen, bilden sie die Grundlage für Diskussionen im Hinblick auf eine Reduktion von Verweildauern und Kosten sowie die Durchsetzung von Qualitätsstandards.

Patienten, die nicht der homogenen Fallgruppe angehören, dürfen nicht nach dem klinischen Pfad behandelt werden. In diesem Fall ist es entweder erforderlich, dass der Pfad für den Patienten so verändert wird, dass er dessen individuelle Bedürfnisse berücksichtigt oder der Patient muss auf einen anderen Pfad gesetzt werden oder die Patientenbehandlung kann sich nicht an einem klinischen Pfad orientieren. Aber nicht nur zu Beginn, sondern auch während der Patientenversorgung kann der Fall eintreten, dass die Behandlung des Patienten gemäß klinischem Pfad nicht (mehr) angebracht ist. Z.B. werden während den Untersuchungen Nebendiagnosen festgestellt, deren Vorkommen im Standardfall nicht vorgesehen ist, und die eine Änderung der Behandlungsplanung erforderlich machen. Da klinische Pfade nicht dieselbe Verbindlichkeit wie medizinische Richtlinien haben, ist in solchen Situationen eine Anpassung der Behandlungsplanung an die spezifischen Bedürfnisse des Patienten möglich und wird als Abweichung oder Varianz vom klinischen Pfad bezeichnet (siehe Kapitel 2.2). Nicht immer sind die Ursachen für eine Abweichung patientenbezogen; oftmals verursachen auch Änderungen in der stationären Planung oder der Organisation im Krankenhaus Varianz von klinischen Pfaden. In jedem Fall muss der Verantwortliche die Gründe angeben, die eine Änderung der Behandlungsplanung rechtfertigen. Dies geschieht im Rahmen der so genannten »Varianzdokumentation«. Die Erfassung von Abweichungen von klinischen Pfaden als Varianzdokumentation ist grundlegend für die kontinuierliche Weiterentwicklung und Verbesserung der Pfade.

2.1.4 Technische Umsetzung klinischer Pfade

Klinische Pfade bedeuten für Krankenhäuser einen radikalen Wandel vom Denken in einer streng hierarchisch geordneten Aufbauorganisation hin zu prozessorientierten Sichten auf die Ablauforganisation. Während die Aufbauorganisation ein Unternehmen nach Aufgaben- und Verantwortungsbereichen gliedert, setzt die Ablauforganisation den Fokus auf die Durchführung der Aufgaben in ihrem zeitlichen und räumlichen Zusammenhang [Becker & Kahn 2005].

Die Vision, nach der sich Unternehmensstrukturen stärker an der Ablauforganisation ausrichten sollen, wurde in der Literatur bereits im Jahr 1934 bei Nordsieck erwähnt [Nordsieck 1934]. Praktische Relevanz erhielt die Thematik erst ab den 1980er Jahren durch Veröffentlichungen von Gaitanides 1983, Porter 1989, Davenport 1993, Hammer & Champy 1993 und neue An-

sätze in der Informationstechnologie, die die Erfassung und Steuerung komplexer Prozessabläufe erst ermöglichten. Die vor allem seit den 1990er Jahren entwickelten Werkzeuge für die Prozessmodellierung, -simulation und -analyse erlangen jedoch erst dann Nachhaltigkeit, wenn die optimierten Prozesse anschließend durch ablauforientierte Anwendungssysteme intelligent und flexibel unterstützt werden; IT-Systeme, die die einzelnen Phasen des Prozesslebenszyklus, von der Modellierung bis zur Ausführung und Analyse unterstützen, werden auch als Workflow Management Systeme (WfMS) bezeichnet.

Die Änderung vom Denken in Aufbauorganisationen hin zum Denken in Ablauforganisationen spiegelt sich also auch in der Konzeption und Realisierung von IT-Systemen wieder. Um die Bedeutung der Prozessorientierung zu verstehen, muss zunächst das bisher geltende Prinzip der Funktionsorientierung thematisiert werden. Auf dieser Grundlage ist es möglich, den Prozessbegriff klar zu definieren und den Beitrag von WfMS zur Prozessunterstützung zu erörtern. Abschließend wird das Potential und die Beschränkungen des Einsatzes solcher Systeme zur Umsetzung klinischer Pfade diskutiert.

2.1.4.1 Neues Paradigma »Prozessorientierung«

Die Orientierung von Organisationen an der Durchführung von Einzelfunktionen hat zur Entwicklung und Optimierung lokaler Funktionsbereiche geführt, wie z. B. Rechnungswesen oder Logistik. Gleichzeitig trat der Gesamtzusammenhang über die betrieblichen Abläufe in den Hintergrund. Mit der zunehmenden Autonomie der Funktionsbereiche stiegen jedoch Aufwand und Kosten für die Koordination zwischen den Bereichen. Da es sich hierbei um ein strukturelles Problem handelt, kann auch der Einsatz moderner Informations- und Kommunikationstechnologien das Problem nicht grundsätzlich beseitigen; stattdessen haben sich die betrieblichen Informationssysteme der Konzentration auf Funktionsbereiche angepasst, so dass die Informationsverarbeitung überwiegend funktions- und datenbezogen stattfindet [Breitbart et al. 1993]. Die Ausführung der Funktionen wird durch die Bereitstellung der notwendigen Daten im jeweiligen Anwendungskontext unterstützt. Die logische Verbindung der Funktionen im Sinne eines Arbeitsprozesses ist hingegen nicht vorgesehen. Mit zunehmender Komplexität der Prozessabläufe steigt daher bei der funktionsorientierten, passiven Form der Datenbereitstellung die Gefahr, dass ausstehende Aufgaben übersehen oder Abhängigkeiten z.B. im Hinblick auf die Verfügbarkeit von Informationen nicht berücksichtigt werden. Die Folgen können Verzögerungen, Unterlassung, Leerlauf, Überlastung, unnötige wiederholte Durchführung von Arbeiten oder falsche Entscheidungen sein. Gerade bei der Patientenversorgung hat dies nicht nur Auswirkung auf die Kostenentwicklung sondern auch auf Qualität, Patientenzufriedenheit und damit auf die Wettbewerbsfähigkeit einer Klinik.

Um den Überblick über die Zusammenhänge und Schnittstellen zwischen den Funktionsbereichen zu erhalten, ist die Einführung eines neuen Paradigmas, nämlich das der Prozessorientierung, notwendig; das bedeutet, dass das Informationssystem die durchzuführenden Aufgaben aufeinander abstimmt und Daten aktiv gemäß dem ihm bekannten Arbeitsprozess für die Verantwortlichen bereitstellt. Nach Davenport handelt es sich bei einem Prozess um

> »eine zeitlich und räumlich spezifisch strukturierte Menge von Aktivitäten mit einem Anfang und einem Ende, sowie klar definierten Inputs und Outputs« [Davenport 1993][4].

Aktivitäten entsprechen einzelnen Arbeitsschritten, die zur Erbringung einer Leistung erforderlich sind. Dieser Definition zufolge repräsentiert bereits eine Verkettung weniger Arbeitsschritte einen Prozess. Soll die gesamte Leistungserbringung einer Organisation betrachtet, gesteuert und optimiert werden, müssen einzelne Prozessteile jedoch im Zusammenhang mit dem Gesamtprozess betrachtet werden, da gerade an den Schnittstellen zwischen den Verantwortungsbereichen die größten Koordinationsprobleme entstehen.

Zur klaren Abgrenzung zwischen isolierten Teilprozessen und übergeordneten, häufig bereichsübergreifenden Prozessen wird in der Managementliteratur der Begriff des »Geschäftsprozesses« eingeführt. Hierbei handelt es sich um einen speziellen Prozess,

> »der der Erfüllung der obersten Ziele der Unternehmung (Geschäftsziele) dient und das zentrale Geschäftsfeld beschreibt« [Becker et al. 2005][5].

Ein wesentliches Merkmal des Geschäftsprozesses ist die Betrachtung der Schnittstellen des Prozesses zu den Marktpartnern des Unternehmens; bei klinischen Pfaden sind das in erster Linie andere Leistungserbringer, Logistikunternehmen, externe Dienstleister und die Patienten des Krankenhauses. Überhaupt weisen Geschäftsprozesse eine starke Kundenorientierung auf. Dementsprechend dürfen nur solche Aktivitäten in den Geschäftsprozess aufgenommen werden, die dazu beitragen, dass die erbrachte Leistung den Anforderungen des Kunden genügt; alle anderen Leistungen sollten eliminiert werden, da sie einen Kostenfaktor darstellen ohne sich positiv auf die Wertschöpfung und die Kundenzufriedenheit auszuwirken. Im Hinblick auf die Wertschöpfung unterscheidet Porter zwischen Kernprozessen und Supportprozessen [Porter 1989][6]. Da sich Kernprozesse aus primären Aktivitäten mit direktem Wertschöpfungsbezug zusammensetzen und damit strategische Bedeutung haben, entsprechen sie der engen Definition von Geschäftsprozessen. Supportprozesse hingegen bestehen aus sekundären Aktivitäten, die nicht direkt zur Wertschöpfung beitragen, jedoch für die Ausführung der Kernprozesse zwingend notwendig sind. Die Aufteilung in Kern- und Supportprozesse kann Unternehmen die Entscheidung über die Ausgliederung (»Outsourcing«) bestimmter Funktionsbereiche erleichtern. Geschäftsprozesse stellen betriebliche Abläufe üblicherweise aus betriebswirtschaftlicher Sicht mit hohem Abstraktionsgrad in einer für alle Anwender verständlichen Form dar. Dementsprechend sind die Prozesse zu unspezifisch, um durch IT-Systeme ausgeführt werden zu können. »Workflow Management Systeme« (WfMS) werden in einer Vielzahl von Publikationen als viel versprechende Technologie zur Umsetzung prozessorientierter Anwendungen charakterisiert [Hsu 1993, Hsu 1995, Jablonski & Bussler 1996, Vossen & Becker 1996, Härder 1997, Reichert & Dadam 2000].

In Abgrenzung zum Geschäftsprozess bezeichnet »Workflow« einen Prozess, dessen Durchführung in der Kontrollsphäre eines Anwendungssystems, des WfMS, liegt [Becker et

[4] Vgl. Davenport 1993, S. 5

[5] Vgl. Becker et al. 2005, S. 6f

[6] Vgl. Porter 1989, S. 63ff

al. 2005][7]. Die Transformation eines Geschäftsprozesses in einen Workflow entspricht daher einem Übergang zwischen Beschreibungs- und Ausführungsebene [Jablonski et al. 1997b]. Im Vergleich mit Geschäftsprozessen weisen Workflows im Allgemeinen eine feinere Granularität und eine größere Anzahl an Attributen auf, die für die Interpretation und Ausführung durch ein WfMS notwendig sind, aber auf der für Fachanwender verständlichen Ebene des Geschäftsprozesses keinen Sinn machen. Das folgende Kapitel behandelt die technische Umsetzung von Geschäftsprozessen und speziell klinischen Pfaden mit Hilfe von Workflow Management Systemen.

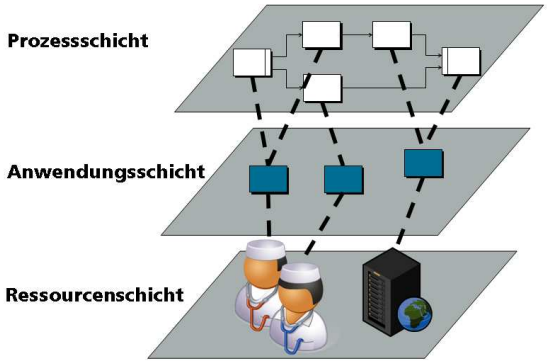
2.1.4.2 Prozessunterstützung durch Workflow Management Systeme

Die stärkere Fokussierung auf Prozesse führt mit Beginn der 1990er Jahre zur Entkopplung der prozessorientierten Geschäftslogik von der daten- und funktionsorientierten Applikationslogik in das WfMS [van der Aalst 1998, van der Aalst & van Hee 2004]. Jablonski et al. spezifizieren einen Workflow als eine zum Teil automatisiert ablaufende Menge von Aktivitäten, die in Zusammenhang mit Teilen eines Geschäftsprozesses oder anderen Vorgängen stehen [Jablonski et al. 1997b][8]. Jeder Workflow besitzt einen definierten Anfang, einen klar organisierten Ablauf und ein definiertes Ende. WfMS dienen der Ausführung von Workflows. Die 1993 von verschiedenen Interessensgruppen aus Forschung und Industrie gegründete Workflow Management Coalition (WfMC)[9] definiert ein WfMS als

> »a system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications« [WfMC 1999][10].

Die nachfolgende Abbildung illustriert den Zusammenhang zwischen Prozesslogik, Anwendungen sowie materiellen und personellen Ressourcen.

Abbildung 4 Entkopplung von Prozessen von Anwendungen und Ressourcen



[7] Vg. Becker et al. 2005, S. 604
[8] Vgl. Jablonski et al. 1997, S. 490
[9] Siehe http://www.wfmc.org
[10] Vgl. WfMC 1999, S. 9

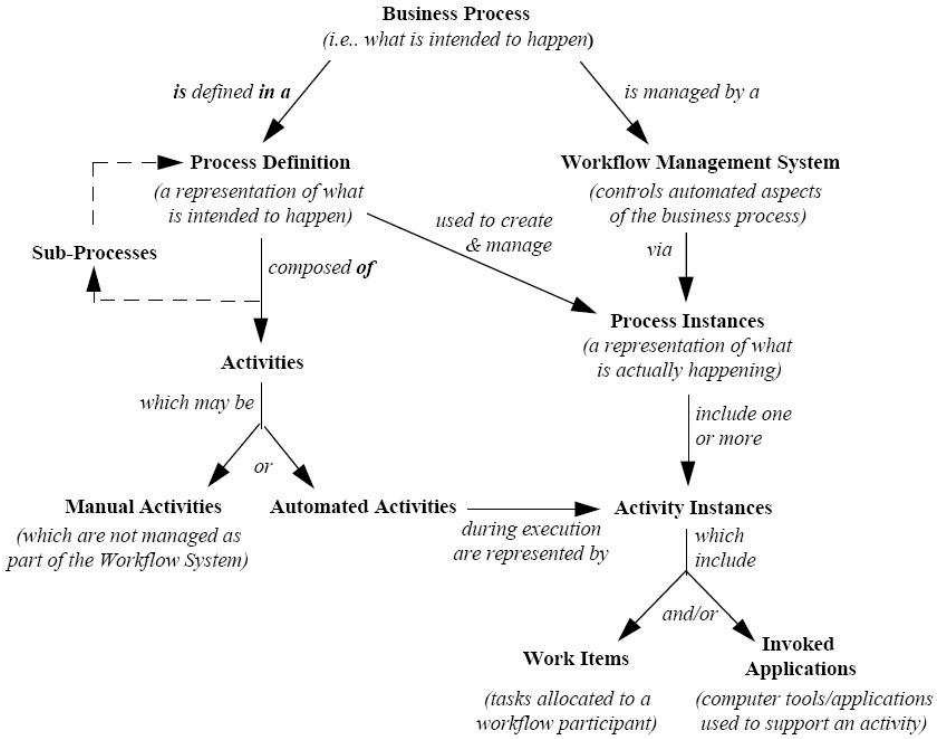
Im Gegensatz zu Geschäftsprozessen sind die Aktivitäten eines Workflows operationalisiert; das bedeutet, sie sind mit Anwendungskomponenten und Datenstrukturen verknüpft, die für die Ausführung der Aktivität notwendig sind. Die Anwendungskomponenten operieren mit Ressourcen, die für die Erfüllung der Aktivität benötigt werden. Die Durchführung einer Aktivität kann z. B. durch menschliches Personal oder durch Maschinen erfolgen; Darüber hinaus können weitere Personen, Systeme oder Räume erforderlich sein.

Durch die getrennte Verwaltung von Prozessdefinitionen von operationalen Anwendungen und Systemen ergibt sich eine Reihe von Vorteilen [van der Aalst 1998]. So können Geschäftsprozesse und die ihnen zugeordneten Workflows unabhängig vom Applikationscode angepasst und gepflegt werden, um zügig auf neue Anforderungen und Änderungen im Geschäftsumfeld zu reagieren; die Auswirkung solcher Modifikationen auf die technische Realisierung des Workflows ist durch klar definierte Schnittstellen zur Anwendungsschicht leicht nachvollziehbar. Die separat vorliegende Prozessdefinition eröffnet zudem Optionen zur Simulation, Überprüfung und Optimierung von Prozessabläufen; auf diese Weise besteht die Möglichkeit nachzuvollziehen, ob ein Prozess oder Teile davon das gewünschte Verhalten zeigen, ohne bereits vollständig operationalisiert zu sein.

2.1.4.2.1 Grundlegende Begrifflichkeiten und Konzepte

In diesem Kapitel werden gemäß den Festlegungen der WfMC grundlegende Begrifflichkeiten und Konzepte in Zusammenhang mit WfM geklärt. Einen Überblick über die Konzepte und ihre Zusammenhänge untereinander gibt Abbildung 5.

Abbildung 5 Überblick über die Konzepte in Zusammenhang mit Workflow Management (Quelle: WfMC 1999, S. 7)



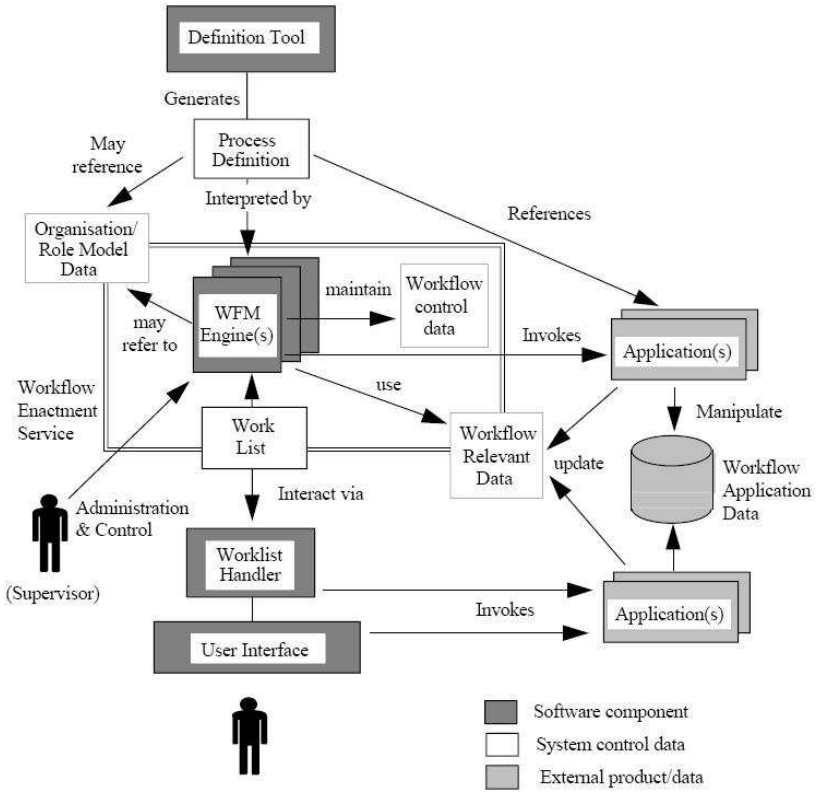
Der Geschäftsprozess entspricht einer Dokumentation betrieblicher Abläufe als Basis für inhaltliche Diskussionen und zur Identifikation von Optimierungspotential. Aus diesem Grund wird der Geschäftsprozess aus Sicht der Domänenexperten (z.B. ärztliches und pflegerisches Personal) beschrieben. Bei der Prozessdefinition bzw. dem Workflow handelt es sich hingegen um eine formale Spezifikation des zuvor entworfenen Geschäftsprozesses, die von einem WfMS ausgeführt werden kann. Die Prozessdefinition besteht aus einer Menge von Aktivitäten, die entweder von einem menschlichen Bearbeiter (manuelle Aktivitäten) oder automatisiert von einer Systemanwendung (automatisierte Aktivitäten) durchgeführt werden. Beispiele für manuelle Tätigkeiten sind das Ausfüllen eines Formulars oder das Erstellen eines Dokuments; bei einer automatisierten Aktivität kann es sich z.B. um das elektronische Versenden einer Rechnung handeln. Zum Zweck der hierarchischen Strukturierung des Prozesses, ist es möglich, Aktivitäten zu Subprozessen zusammenzufassen, die anschließend in die Prozessdefinition eingebettet werden. Optional kann der Subprozess auch selbst eine gültige und instanziierbare Prozessdefinition sein.

Soll der Prozess für einen bestimmten Fall durchlaufen werden, so wird er durch das WfMS instanziiert. Bei einer Prozessinstanz handelt es sich also um die Darstellung einer einzelnen Ausführung einer Prozessdefinition. Jede Prozessinstanz erhält eine innerhalb des Systems eindeutige Kennung und sämtliche Daten werden separat für diese Instanz verwaltet. Der Status der Prozessinstanz orientiert sich an der Ausführung ihrer Aktivitäten und Subprozesse und richtet sich damit nach dem Fortschritt der Bearbeitung. Analog zur Instanziierung der Prozessdefinition wird bei jedem Aufruf einer Aktivität eine Aktivitäteninstanz vom WfMS erzeugt und verwaltet, bis ihre Ausführung beendet ist. Welche Status Aktivitäten- und Prozessinstanzen annehmen können, richtet sich nach der Ausführungssemantik des jeweiligen WfMS.

Mit der Instanziierung einer Aktivität werden ein oder mehrere Arbeitseinheiten (engl. »Work Items«) erstellt, die durch einen zuständigen, meist menschlichen Bearbeiter durchgeführt werden. Für die Darstellung der Arbeitseinheiten für den Bearbeiter werden häufig so genannte Arbeitslisten (engl. »Worklists«) eingesetzt. Kommen mehrere Bearbeiter in Frage, wird jedem von ihnen die Aktivität in seiner Arbeitsliste angeboten; sobald die Aktivität von einem Bearbeiter ausgewählt wurde, wird sie aus den Arbeitslisten der anderen Bearbeiter entfernt. Die Verwaltung der Arbeitslisten erfolgt durch eine entsprechende Arbeitslistenverwaltungskomponente (engl. »Worklist Handler«). Alternativ können Aktivitäten auch direkt mit der Durchführung bestimmter Applikationen (z.B. zur automatischen Auftragsgenerierung) (»Invoked Applications«) verknüpft sein.

Üblicherweise unterscheidet man in Zusammenhang mit WfMS zwischen Modellierungs- und Laufzeitkomponenten. Einen Überblick über die einzelnen Komponenten und ihre Beziehungen untereinander gibt Abbildung 6. In den Folgenden Kapiteln werden die Funktionen der Komponenten näher beschrieben.

Abbildung 6 Übersicht über die Struktur von WfMS (Quelle: Hollingsworth 1995, S. 13)



2.1.4.2.2 Modellbezogene Aspekte

Die Entwicklung eines prozess-orientierten Anwendungssystems beginnt mit der Überführung eines Geschäftsprozesses in ein formales, durch IT-Systeme lesbares und interpretierbares Modell, die Prozessdefinition. Für die Entwicklung von Modellen benötigt man Informationen über die Konstrukte, die bei der Modellierung zur Verfügung stehen und die Beziehungen, die zwischen den Konstrukten möglich sind. Diese Informationen stammen aus dem durch das WfMS bereitgestellten WF-Metamodell, das definiert, welche Konstrukte zur Beschreibung verschiedener Workflow Aspekte verfügbar sind. Grundsätzlich werden in der Literatur fünf Aspekte unterschieden, die von jedem WfMS unterstützt werden sollten [Jablonski & Bussler 1996]:

- Der funktionale Aspekt definiert, welche Tätigkeiten im Rahmen eines Prozesses ausgeführt werden. Gleichzeitig legt er die hierarchische Anordnung von Aktivitäten in Subprozesse fest.
- Der verhaltensbezogene Aspekt beschreibt die Reihenfolge, in der Tätigkeiten ablaufen. Zu diesem Zweck werden Aktivitäten oder Subprozesse durch Kontrollflüsse miteinander verbunden.
- Der informationsbezogene Aspekt gibt an, welche Informationen und Dokumente im Rahmen der Prozessausführung konsumiert und produziert werden. Mit Hilfe des Datenflusses

wird spezifiziert, welche Aktivitäten lesend oder schreibend auf Daten zugreifen. Die WfMC unterscheidet zwischen drei Kategorien von Daten [WfMC 1999][11]: Kontrolldaten liegen ausschließlich in der Einflusssphäre des WfMS, z. B. Statusinformationen zu Prozess- und Aktivitätsinstanzen. Im Gegensatz dazu werden Anwendungsdaten allein von der jeweiligen Applikation verwaltet und sind für das WfMS nicht zugreifbar. Workflow relevante Daten können zwischen dem WfMS und den Anwendungskomponenten ausgetauscht werden; sie werden z.B. verwendet, um über die weitere Ausführung bei bedingten Verzweigungen zu entscheiden.

- Der organisatorische Aspekt beschreibt, welche Ressourcen benötigt werden, um bestimmte Tätigkeiten auszuführen. Bei Ressourcen kann es sich z.B. um Personen, Systeme oder Räume handeln. Die Zuordnung von Ressourcen zu Aktivitäten kann organisatorischen Regeln unterliegen, die z. B. mit Zugriffsberechtigungen oder Vertretungsregelungen assoziiert sind. Um definieren zu können, welche Bearbeiter für die Ausführung einer bestimmten Tätigkeit in Frage kommen, werden meist Rollenkonzepte hinterlegt; einige WfMS gestatten es darüber hinaus, Bearbeiter organisatorischen Einheiten zuzuordnen.
- Im Rahmen des operationalen Aspekts wird festgelegt, wie Aktivitäten geeignet mit Anwendungskomponenten gekoppelt werden. Der operationale Aspekt entspricht dem Übergang von der Prozessschicht zur Anwendungsschicht (siehe Abbildung 4). Dabei muss auch festgelegt werden, wie Eingabeparameter vom Datenkontext des WfMS an die Anwendung übergeben und wie nach Beendigung der Aktivität die Ausgabeparameter der Anwendung in den Datenkontext des WfMS übernommen werden (Workflow relevante Daten). Prinzipiell können die Anwendungskomponenten unabhängig vom WfMS entwickelt und gepflegt werden; häufiger sind jedoch Anpassungen von bereits existierenden Komponenten (so genannte »Legacy Systeme«) notwendig, da ursprünglich eine Kommunikation mit einem WfMS nicht vorgesehen war. Beispiele für Anwendungskomponenten sind Benutzeroberflächen, die die Interaktion mit einem menschlichen Bearbeiter steuern, oder komplette Programme. Die Auswahl einer geeigneten Anwendungskomponente kann entweder statisch zur Modellierungszeit oder dynamisch zur Laufzeit erfolgen.

Für eine ausführliche Diskussion dieser und anderer Aspekte sei an dieser Stelle auf die weiterführende Literatur verwiesen [Jablonski & Bussler 1996].

Im Referenzmodell der WfMC dient die Komponente »Definition Tools« als Werkzeug zur Erstellung von Prozessdefinitionen. Die Mehrzahl aktueller WfMS bietet zur Entwicklung von Prozessdefinitionen graphische Beschreibungssprachen an, die der visuellen Repräsentation der im WF-Metamodell definierten Konstrukte dienen. Aktivitäten werden z. B. häufig durch Rechtecke, Kontrollflüsse durch gerichtete Pfeile symbolisiert. Je ausdrucksmächtiger das zugrunde liegende WF-Metamodell ist, desto komplexere Prozesse können abgebildet werden. Mit den so genannten »Workflow Patterns« wurde eine Menge von Konstrukten definiert, an Hand derer die Ausdrucksmächtigkeit von Metamodellen im Hinblick auf den Kontrollfluss überprüft werden kann [van der Aalst et al. 2003, Russell et al. 2006a]. Ähnliche Arbeiten existieren für den Datenfluss (»Workflow Data Patterns«) [Russell et al. 2004a], die Zuordnung von Ressourcen zu Aktivitäten (»Workflow Resource Patterns«) [Russell et al. 2004b] und für die Behandlung von Ausnahmefällen (»Exception Handling Patterns«) [Russell et al. 2006b].

[11] Vgl. WfMC 1999, S. 44-46

Nach der Fertigstellung und Operationalisierung der Prozessdefinition kann die Instanziierung und Ausführung des Prozesses beginnen. Die meisten WfMS stellen für diesen Zweck standardisierte Benutzeroberflächen bereit sowie Programmierschnittstellen, um die Standardoberflächen durch spezifische Oberflächen zu ersetzen oder um die Funktionalität des WfMS in bestehende Systeme zu integrieren.

2.1.4.2.3 Ausführungsbezogene Aspekte

Prozessdefinitionen, die mit Hilfe eines geeigneten Modellierungswerkzeugs erstellt wurden, können anschließend durch die Laufzeitkomponente »Workflow Engine« instanziiert und ausgeführt werden. Zu diesem Zweck wählt ein Berechtigter die Prozessdefinition aus dem Repository, in dem sämtliche Prozessdefinitionen gespeichert sind, zur Instanziierung aus. Von jeder Prozessdefinition können Prozessinstanzen unabhängig voneinander erzeugt werden. Prozessinstanzen verfügen über eigene Laufzeitparameter und Ausführungsstatus. Üblich ist eine Unterscheidung zwischen Zuständen wie »initiiert«, »laufend«, »aktiv«, »unterbrochen«, »abgeschlossen«, »abgebrochen« und »archiviert« [WfMC 1999][12]. Analog zur Prozessinstanz werden auch die Status der Aktivitätsinstanzen verwaltet; mögliche Zustandswerte sind »inaktiv«, »aktiv«, »unterbrochen« und »abgeschlossen« [WfMC 1999][13]. Sämtliche Funktionen der Workflow Engine werden gemäß WfMC Architektur durch die Komponente »Workflow Enactment Service« gekapselt; auf diese Weise ist es möglich, mehrere verteilt arbeitende Workflow Engines für ein WfMS zu nutzen [Hollingsworth 1995][14].

Die Workflow Engine steuert und überwacht die Ausführung der Prozessinstanzen während ihrer gesamten Lebensdauer; sie verwaltet die instanzspezifischen Daten, trägt manuelle Aktivitäten in die Arbeitslisten möglicher Bearbeiter ein und führt die automatisierten Aktivitäten selbständig aus, indem sie die entsprechenden Anwendungskomponenten ansteuert. Jablonski & Bussler [Jablonski & Bussler 1996] definieren zu diesem Zweck eine »Workflow Applikation«, die als Schnittstelle zwischen der Aktivität innerhalb der Prozessdefinition und der konkreten Implementierung der Anwendung fungiert. Als Technologie für die Umsetzung dieser Schnittstelle werden inzwischen hauptsächlich Web Services genutzt; das WfMS wird in diesem Zusammenhang als Bestandteil einer übergeordneten Service-orientierten Architektur (SOA) betrachtet [Sanjiva et al. 2005].

2.1.4.3 Potentiale und Grenzen von WfMS

Die Probleme, die mit einer vorwiegend daten- und funktionsbezogenen elektronischen Informationsverarbeitung verbunden sind sowie die Notwendigkeit einer bereichsübergreifenden, qualitativ hochwertigen und kosteneffizienten Gestaltung der Behandlungsabläufe, haben dazu geführt, dass das Interesse von Krankenhäusern an einer IT-basierten Prozessteuerung in den letzten Jahren enorm zugenommen hat. Dementsprechend wird der Einsatz von WfMS zur Umsetzung klinischer Pfade in der Literatur vielfach diskutiert [Reichert et al. 1997, Dadam

[12] Vgl. WfMC 1999, S. 47
[13] Vgl. WfMC 1999, S. 48
[14] Vgl. Hollingsworth 1995, S. 13-14

et al. 2000, Reichert 2000a, Lenz & Kuhn 2004, Lenz & Reichert 2007]. Es wird festgestellt, dass WfMS hohes Potential bergen, um Behandlungsprozesse in Krankenhäusern effizienter und effektiver zu steuern und damit ihre Wirtschaftlichkeit und Wettbewerbsfähigkeit zu erhöhen. Im Folgenden werden einige aus Krankenhaussicht vielversprechende Perspektiven aufgelistet:

- Da die Prozessdefinition Verantwortlichkeiten festschreibt und Schnittstellen zwischen Fachpersonal und Abteilungen explizit aufzeigt, können Abstimmungsschwierigkeiten an den Übergängen verhindert werden.
- Durch eine adäquate Koordination der im Behandlungsprozess anstehenden Tätigkeiten und die rechtzeitige Übermittlung der für die Durchführung benötigten Dokumente, ist es möglich, einerseits die Effizienz der Bearbeitung zu erhöhen und andererseits Warte- und Leerlaufzeiten zu reduzieren.
- Arbeitslisten für Fachpersonal können rollen- und anwenderspezifisch verwaltet werden, um einen schnellen Überblick über die für den Bearbeiter notwendigen Tätigkeiten zu geben; darüber hinaus ist auch eine Sortierung von Arbeitseinheiten nach definierten Parametern, wie z.B. Dringlichkeit oder Aktivitätstyp, möglich.
- Die aktive Kontrolle des Fortschritts der Prozessausführung durch das WfMS erleichtert das frühzeitige Erkennen von Unterlassungsfehlern, die z.B. eine Verlängerung der Liegezeit von Patienten zur Folge haben können.
- Anwendungskomponenten können automatisch in Abhängigkeit der aktuell auszuführenden Aktivität z.B. zur Auftragsvergabe an die Radiologie oder das Labor, angesteuert werden; auf diese Weise kann die administrative Arbeitslast der Mitarbeiterinnen und Mitarbeiter reduziert werden.
- Behandlungsverlaufe werden nachvollziehbarer, indem WfMS Funktionen zur Abfrage des Ausführungsstatus, Einsehen der Behandlungshistorie und der noch ausstehenden Aufgaben bereitstellen.
- WfMS unterstützen die prozessorientierte und vollständige medizinische Dokumentation des Behandlungsprozesses.
- Die Kostenkontrolle wird durch die Zuordnung von Kostenstellen zu Geschäftsprozessen (z.B. DRGs zu klinischen Pfaden), zu Aktivitäten und Subprozessen (z.B. Personal- und Materialkosten) vereinfacht.
- WfMS lassen sich als Bestandteil einer SOA in die oftmals sehr heterogene Systemlandschaft von Krankenhäusern integrieren; statt den Austausch existierender und bewährter Anwendungssysteme erforderlich zu machen, ermöglichen sie die prozessorientierte Verknüpfung solcher Systeme.
- Durch das Vorliegen separater Prozessdefinitionen können Abläufe durchgespielt und IT-gestützt simuliert werden, ohne dass die Anwendungskomponenten vollständig realisiert sind.
- Die Auswirkungen von Änderungen an Prozessdefinitionen auf Anwendungskomponenten lassen sich aufgrund der klar definierten Schnittstellen leichter abschätzen, als wenn die Prozessdefinition nicht explizit vorliegt, sondern die Ablauflogik im Programmcode der Anwendungssysteme verborgen ist.

Obwohl der Einsatz von WfMS im klinischen Bereich deutliche Vorteile für die effiziente und effektive Gestaltung des gesamten Behandlungsprozesses hat, konnte sich die Technologie in Krankenhäusern bisher kaum durchsetzen. Eine wesentliche Ursache hierfür besteht in der »Befürchtung des Personals, in der Tätigkeit zu stark eingeschränkt zu werden und somit den individuellen Behandlungsbedürfnissen der Patienten nicht mehr nachkommen zu können« [Kleemann 2010]. Diese Befürchtung ist auch nicht unberechtigt, wirft man einen Blick auf die

Besonderheiten klinischer Prozesse, die der Anwendbarkeit konventioneller WfM-Technologie in Krankenhäusern Grenzen setzen:

- Die Komplexität medizinischer Behandlungsabläufe erschwert die Darstellung klinischer Pfade mit den Mitteln aktueller Modellierungswerkzeuge; je nach Behandlungskontext existiert eine Menge alternativer Behandlungswege, deren Berücksichtigung Prozessdefinitionen für Anwender zunehmend unübersichtlich machen.
- Die Optionalität von Aktivitäten, Kontroll- und Datenflüssen erschwert die Modellierungsarbeit und die Anbindung geeigneter Anwendungskomponenten zusätzlich. Während der medizinischen Behandlung gibt es eine Reihe von Tätigkeiten die nur bedingt ausgeführt werden, wie z.B. Röntgenthorax bei Patienten, die älter als 50 Jahre sind, oder Anfertigung eines Belastungs-EKGs, wenn dies nicht bereits am Aufnahmetag stattgefunden hat.
- Fachexperten ohne fundierte technische Kenntnisse können die aktuellen Prozessmodellierungswerkzeuge gar nicht bedienen, um technisch ausführbare Prozessdefinitionen zu erzeugen; sie sind daher stets auf IT-Spezialisten angewiesen. Dies erhöht den Zeit- und Arbeitsaufwand gerade in einer dynamischen Domäne mit häufigen Änderungszyklen überproportional.
- Aufgrund der individuellen Gesundheitsbedingungen und Unterschieden im sozioökonomischen Umfeld der Patienten sowie organisatorischen Änderungen (z.B. in Folge von Notfällen) ist die Planbarkeit und Vorhersehbarkeit klinischer Behandlungsabläufe eingeschränkt. Das bedeutet, dass Anpassungen an laufenden Prozessinstanzen des klinischen Pfades notwendig sind, um adäquat auf die geänderten Anforderungen reagieren zu können.

2.1.5 Anforderungen an das Lösungskonzept

Damit WfMS flächendeckend zur Umsetzung von klinischen Pfaden eingesetzt werden, muss sich ihr Nutzen relativ frühzeitig in der Aufwandsreduktion zur Erstellung klinischer Pfade widerspiegeln. Wie bereits in der Einleitung dieser Arbeit geschildert wurde, ist die Modellierung von Prozessen allgemein und damit auch die Erstellung klinischer Pfade mit erheblichem Zeit- und Kostenaufwand verbunden. Sollen klinische Pfade auf Basis von WfMS realisiert werden, muss das IT-Management über die notwendigen Kenntnisse bezüglich Entwicklung und Ausführung der Prozessdefinitionen verfügen, da dem pflegerischen und ärztlichen Personal der Umgang mit den stark technisch orientierten Werkzeugen nicht zugemutet werden kann. Aufgrund der Komplexität medizinischer Abläufe und des erforderlichen fachlichen Know-Hows, ist es für IT-Spezialisten jedoch schwierig, die Prozessdefinitionen so zu erstellen, dass sie den Anforderungen des medizinischen Personals gerecht werden. Sind die Prozessdefinitionen unzureichend oder fehlerhaft spezifiziert, haben Ärzte und Krankenschwestern auch keine Möglichkeit, den klinischen Pfaden selbst zu modifizieren, sondern müssen alle notwendigen Schritte stets mit dem IT-Management absprechen. Daraus ergibt sich die erste Anforderung an das Lösungskonzept zur prozessorientierten Umsetzung klinischer Pfade:

Anforderung: Das Lösungskonzept muss es Fachexperten ermöglichen, klinische Pfade selbst zu modellieren.

Obwohl der Aufwand zur Erstellung klinischer Pfade mit zunehmender Erfahrung abnimmt, kann auch das WfMS dazu beitragen, die Effizienz bei der Erstellung neuer Prozessdefinitionen zu steigern. Dazu müssen alle Personen die an der Entwicklung der Pfade beteiligt sind, konsequent dazu angehalten werden, auf vorhandene Pfadkomponenten zuzugreifen, diese direkt zu nutzen oder an die aktuellen Erfordernisse anzupassen. Die Modellierung muss intuitiv und ohne weitreichende IT-Kenntnisse stattfinden können.

Anforderung: Das Lösungskonzept muss formale Verfahren anbieten, die dazu beitragen, das entstandene Prozesswissen zu verstetigen und bei der Erstellung neuer Pfade konsequent nutzbar zu machen.

Die meisten WfMS stellen bereits Werkzeuge zur Verfügung, um bestimmte Prozessteile als wieder verwendbare Komponenten zu kennzeichnen und sie dann z. B. in Form von Subprozessen in unterschiedlichen Prozessdefinitionen zu referenzieren. Eine konsequente Nutzung bereits erstellter Komponenten ist jedoch nur möglich, wenn sie in einen Nutzungskontext gesetzt werden und Zusammenhänge und wechselseitige Beziehungen zwischen ihnen eindeutig definiert werden können.

Anforderung: Das Lösungskonzept muss Möglichkeiten bereitstellen, Nutzungskontexte für wieder verwendbare Komponenten zu definieren und semantische Beziehungen zwischen den Komponenten auszudrücken. Das Konzept muss so generisch sein, dass es bei Bedarf um weitere Beziehungstypen ergänzt werden kann. Die Korrektheit von Beziehungsnetzen muss formal nachweisbar sein.

Im Krankenhaus gibt es viele Aktivitäten, die in nahezu identischer Form zu unterschiedlichen Zeiten und damit an unterschiedlichen Stellen der Prozessdefinition vorkommen. Der Aufwand zur Modellierung klinischer Pfade kann daher erheblich gesenkt werden, wenn die Option zur gleichzeitigen und gleichartigen Entwicklung unterschiedlicher Teile der Prozessdefinition besteht.

Anforderung: Das Lösungskonzept muss Methoden bereitstellen, um durch die gleichzeitige und gleichartige Ausgestaltung unterschiedlicher Bereiche innerhalb der Prozessdefinition, den Entwicklungsaufwand deutlich zu senken.

Da die Bedeutung von Fachexperten gemäß dem anzustrebenden Lösungskonzept weit über ihre bisherigen Rollen als Informationsquelle für IT-Spezialisten und Validierungsinstanz von Prozessdefinitionen hinausgeht, muss das Konzept inhärent auf die Überführung klinischer Pfade in unterschiedliche Repräsentationsformen ausgerichtet sein. Die fachliche Darstellung klinischer Pfade für das medizinische Personal kann sich z. B. wesentlich von der Repräsentation von IT-Spezialisten unterscheiden. Dabei sei anzumerken, dass das Konzept lediglich Verfahrensweisen zur Transformation der Prozessdefinition anbieten muss; es ist nicht Gegenstand der Arbeit domänenspezifische Modellierungssprachen zu entwickeln – vielmehr soll die Verwendung unterschiedlicher existierender Sprachen vereinfacht werden.

Anforderung: Das Lösungskonzept muss ein Verfahren definieren, nachdem Prozessdefinitionen möglichst einfach und kontrolliert in unterschiedliche Zielsprachen überführt werden können.

WfMS können nicht ihren vollen Mehrwert für das Behandlungsmanagement entfalten, solange klinische Pfade lediglich als Prozessdefinitionen vorliegen, an denen sich die Patientenversorgung zu orientieren hat, die jedoch nicht ausgeführt werden können. Die Prozessausführung ist mit der Entwicklung von Anwendungskomponenten verbunden, wie z. B. Web Services, Programmdateien oder Benutzerschnittstellen, die aufgerufen werden, sobald der Prozess einen definierten Ausführungsstatus erreicht. Die klinischen Pfade, die aus Sicht der Fachexperten spezifiziert werden, sollen sofort nach ihrer Erstellung für Patienten instanziier- und ausführbar sein. Nur so kann die semantische Korrektheit der Modelle überprüft und die Prozessdefinitionen bei Bedarf in einem frühen Entwicklungsstadium überarbeitet werden. Die

Ausführbarkeit einer Prozessdefinition setzt die strukturelle Integrität des Kontroll- und Datenflusses voraus. Bereits auf Konzeptebene sollen daher Methoden definiert werden, die dazu führen, dass die bei der Modellierung resultierenden Prozessdefinitionen nach Möglichkeit korrekt sind. Die Korrektheit lässt sich zusätzlich auch dann formal nachweisen, wenn die Definition in eine Prozessmodellierungssprache überführt wird, die bereits Korrektheitskriterien definiert, nach denen z. B. Verklemmungen oder Endlosschleifen ausgeschlossen sind. Da die Systemlandschaft im Gesundheitswesen sehr heterogen ist, muss das Konzept so generisch gehalten werden, dass die Ausführung klinischer Pfade prinzipiell unter Nutzung unterschiedlicher Prozessmodellierungssprachen und WfMS möglich ist.

Anforderung: Klinische Pfade, die von Fachexperten modelliert oder angepasst wurden, müssen sofort unter Nutzung unterschiedlicher WfMS ausführbar sein. Ausführbarkeit bedeutet auch, dass das Konzept Möglichkeiten bieten muss, um zu verhindern, dass Prozessinstanzen in undefinierte und fehlerhafte Zustände geraten, die unkontrollierte Systemabbrüche und Datenverlust zur Folge haben können.

WfMS wurden ursprünglich für die Steuerung industrieller Produktionsprozesse entwickelt. Diese Prozesse sind im Gegensatz zu klinischen Behandlungsabläufen übersichtlich, stark strukturiert und gut planbar. Abweichungen oder Ausnahmesituationen treten nur selten auf. Um WfMS auch in Krankenhäusern erfolgreich einsetzen zu können, bedarf es einer genauen Feststellung, was durch klinische Pfade abgebildet werden soll und was einer Abweichung vom Pfad entspricht. Darüber hinaus müssen Werkzeuge entwickelt werden, die dazu beitragen, Flexibilisierungsmöglichkeiten vor und während der Prozessausführung zu schaffen. Dies erfordert jedoch eine detaillierte Analyse dessen, was überhaupt Varianz von klinischen Pfaden bedeutet.

2.2 Varianz von klinischen Pfaden

Um bestimmen zu können, welche Anforderungen das Lösungskonzept erfüllen muss, um klinische Pfade als Behandlungsstandards im Sinne der individuellen Patientenanforderungen zu flexibilisieren, wird in diesem Kapitel das Phänomen der Varianz vom klinischen Pfad untersucht und an Hand einer Taxonomie klassifiziert. Auf Basis dieser Taxonomie lassen sich auch Anforderungen ableiten, die speziell den geeigneten Umgang mit Varianz adressieren.

Mit Blick auf die Nutzung klinischer Pfade als Basis für die stationäre Behandlung im Krankenhaus können Patienten grob in zwei Klassen unterteilt werden: »Pfad-Patienten« erfüllen die mit dem klinischen Pfad assoziierten Einschluss- und Ausschlusskriterien und zählen damit zur homogenen Gruppe, die gemäß Pfad behandelt werden kann. Für »Nicht Pfad-Patienten« hingegen kommt der Einsatz klinischer Pfade nicht in Frage, sondern die Behandlungsplanung erfolgt individuell in Abhängigkeit der spezifischen Gesundheitssituation des Patienten. Obgleich ein Patient zu Beginn der Behandlung zu der Gruppe der Pfad-Patienten zählt und der klinische Pfad für ihn instanziiert werden kann, besteht die Möglichkeit, dass sich die Behandlungssituation im Laufe der Versorgung so ändert, dass eine strikte Befolgung des Pfades nicht mehr möglich ist, z.B. weil sie negative Auswirkungen auf den Gesundheitszustand des Patienten haben würde. Insgesamt können die Pfad-Patienten daher noch drei unterschiedlichen Kategorien zugeordnet werden:

- **Pfad-Durchläufer:** Diese Gruppe von Patienten entspricht dem Optimalfall, da sie exakt gemäß dem zuvor geplanten Prozessablauf behandelt werden kann. Ihre Zahl ist erfah-

rungsgemäß mit der initialen Einführung klinischer Pfade im Krankenhaus eher gering, nimmt aber im Zuge der Verbesserung und Erweiterung der Pfade stetig zu.
- **Pfad-Abweichler:** Diese Gruppe erfüllt zwar grundsätzlich die Voraussetzungen, um nach dem klinischen Pfad behandelt zu werden, die strikte Einhaltung der Vorgaben des Pfades ist jedoch aus irgendwelchen Gründen nicht möglich. Trotz Abweichungen kann der klinische Pfad aber weiterhin als Grundlage für die Behandlung eingesetzt werden.
- **Pfad-Abbrecher:** Diese Gruppe von Patienten wird initial auf den klinischen Pfad gesetzt; während der Versorgung wird jedoch festgestellt, dass die Behandlung gemäß klinischem Pfad für die Patienten nicht angemessen ist. In diesen Fällen kann der Pfad nicht als Basis für die Organisation und Durchführung der Patientenbehandlung dienen, sondern die Ausführung muss komplett abgebrochen werden.

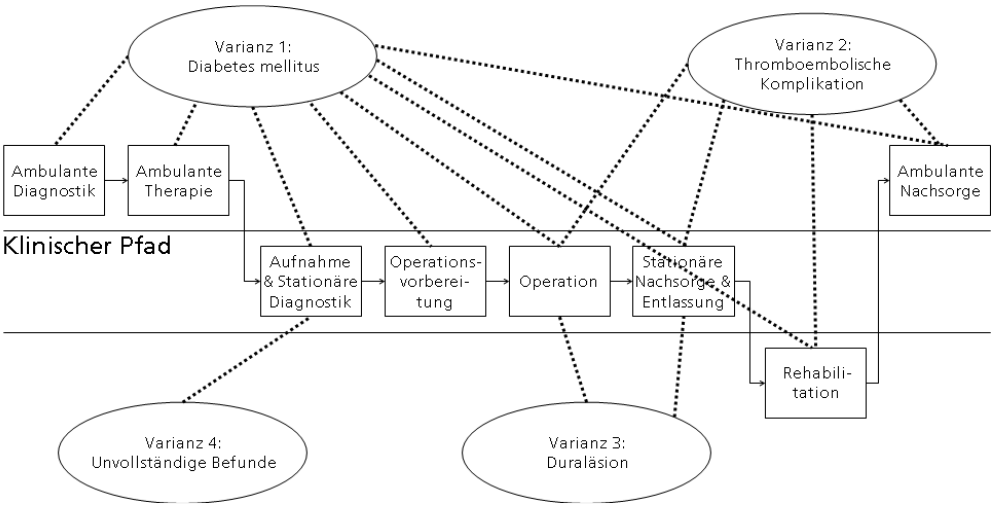
Das Auftreten einer Abweichung der Behandlungssituation, die auch als Varianz bezeichnet wird, kann also zu einer Änderung oder sogar zum Abbruch von Prozessinstanzen klinischer Pfade führen. Um die Gründe und Auswirkungen der Varianz im Hinblick auf die Patientenversorgung nachvollziehen zu können, ist das verantwortliche Personal meist dazu verpflichtet, beides als Varianzdokumentation zu erfassen. Ist das zugrunde liegende WfMS nicht in der Lage, die Auswirkungen der Varianz im Prozessablauf entsprechend zu berücksichtigen, kann das WfMS ausschließlich verwendet werden, um geplante, aber nicht real durchgeführte Prozesse abzubilden. Dies hat zur Folge, dass das ärztliche und pflegerische Personal gezwungen ist, am System vorbeizuarbeiten; einige der wesentlichen Vorteile von WfMS, wie z.B. die Unterstützung einer durchgängigen medizinische Dokumentation, die Erleichterung der Kostenkontrolle und die Koordination der Patientenbehandlung kommen damit nicht mehr zum tragen. Unter den eingeschränkten Nutzungsmöglichkeiten des Systems leidet nicht nur sein Wert für das Krankenhausmanagement, sondern auch seine Akzeptanz durch die Mitarbeiterinnen und Mitarbeiter. Für die Durchsetzung von WfMS in der klinischen Domäne ist die Bereitstellung von Möglichkeiten zum adäquaten Umgang mit Pfadvarianz also eine fundamentale Voraussetzung.

Um die Anforderungen an WfMS beschreiben zu können, ist es notwendig, das Phänomen der Varianz genau zu analysieren. Im Folgenden werden zunächst einige praktische Beispiele für Varianz angeführt, an Hand derer die Unterschiedlichkeit von Varianz und ihre Bedeutung im Rahmen medizinischer Behandlungsprozesse herausgestellt werden. Anschließend erfolgt die Klärung grundlegender Begrifflichkeiten. Auf Basis dieser Erörterungen werden die unterschiedlichen Arten von Varianz in Form einer Taxonomie klassifiziert.

2.2.1 Varianz am Beispiel der stationären Behandlung von Wirbelsäulenerkrankungen

Während Abweichungen von geplanten Prozessabläufen in der Literatur zu WfMS meist als Ausnahmefälle betrachtet werden, ist Variabilität bei der Patientenversorgung fester Bestandteil des klinischen Praxisalltags. Das in diesem Kapitel geschilderte Beispiel soll die Vielfalt und Bedeutung von Pfadvarianz verdeutlichen. Die nachfolgende Abbildung gibt einen Überblick über die stationäre Behandlung von Wirbelsäulenerkrankungen und einiger möglicher Varianzen, die eine Flexibilisierung bei der Prozessausführung erfordern.

Abbildung 7 Überblick über die stationäre Behandlung von Wirbelsäulenerkrankungen



Der Prozess ist nicht detailliert dargestellt, sondern aus Gründen der besseren Übersichtlichkeit in Phasen unterteilt. Die Verbindungen zwischen Varianz und Phase machen ihren Einfluss auf den Prozess deutlich.

2.2.1.1 Prozessbeschreibung

Wirbelsäulenerkrankungen können äußerst vielfältig sein; dazu zählen z. B. chronische Rückenschmerzen, Bandscheibenvorfall, Facettensyndrom, Wirbelkanalverengung, etc. Die meisten Krankheitserscheinungen ereignen sich in Folge altersbedingter Degenerationserscheinungen; beim Bandscheibenvorfall kommt es z. B. zu einer schmerzhaften Verlagerung von Bandscheibengewebe (speziell des Gallertkerns der Bandscheibe) in den Rückenmarkskanal [Fardon 2001]. Der wesentliche Behandlungsprozess gestaltet sich unabhängig von der spezifischen Art der Erkrankung jedoch weitgehend identisch.

Zum Zeitpunkt der stationären Aufnahme kann mit Ausnahme von akuten Fällen davon ausgegangen werden, dass der Patient bereits in ambulanter Behandlung war. Die meisten Wirbelsäulenerkrankungen werden zunächst konservativ behandelt, d. h. Ziel sind die Schmerzbehandlung und die Wiederherstellung der Funktionen des Bewegungsapparates. Die konservative Wirbelsäulentherapie umfasst die vorübergehende entlastende Lagerung des Patienten (z.B. Stufenlagerung), Wärmeanwendungen, die Gabe schmerzlindernder Medikamente, Injektionsbehandlung sowie die kontrollierte Mobilisierung [Krämer 2006][15]. Zeigen die konservativen Maßnahmen auch nach konsequentem Einsatz kein befriedigendes Resultat, so wird vor der offenen Operation meist ein minimal-invasives Behandlungsverfahren gewählt. Erst in Fällen starker Beeinträchtigung der Lebensqualität z.B. durch motorische und sensorische Ausfälle oder bei unzulänglichem Erfolg durch andere Therapiemethoden kann eine Indikation zum operativen Eingriff gegeben sein. Die Operation findet im Krankenhaus statt. Im Vorfeld der Operation erfolgt die Anamnese (Erfassung der Krankenhistorie) und die klinische Unter-

[15] Vgl. Krämer 2006, S. 220-221

suchung (Feststellung der Symptome) statt. Sofern die notwendigen Befunde, z. B. Laborbefunde und Bild gebende Befunde noch nicht vorliegen, müssen diese am Aufnahmetag nachgeholt werden. Die Bild gebenden Methoden, wie z.B. Röntgenuntersuchung, Magnetresonanztomographie (MRT) und Computertomographie (CT), werden in Abhängigkeit der konkreten klinischen Symptomatik angewandt. Zu den weiteren diagnostischen Maßnahmen zählen neben Labortests bei Bedarf auch nuklearmedizinische Verfahren oder neurologische Untersuchungen.

Die häufigste Operationsform ist ein Eingriff an der Bandscheibe, bei der das verlagerte Bandscheibengewebe entfernt wird, um die Nervenwurzel zu entlasten (Diskotomie). Gemäß der Leitlinie »Rehabilitation bei Bandscheibenvorfall mit radikulärer Symptomatik und nach Bandscheibenoperation« der AWMF (Arbeitsgemeinschaft der Wissenschaftlichen Medizinischen Fachgesellschaften) kommen meist folgende Methoden zur Anwendung [Greitemann & Stein 2005]:

- Diskotomie in offener chirurgischer Technik
- Diskotomie in mikrochirurgischer Technik
- Perkutan automatisierte Diskotomien
- Chemonukleolyse
- Laserdiskotomie

Andere Verfahren wie z. B. die Erweiterung des Wirbelkanals oder die Wirbelsäulenversteifung erfolgen in Abhängigkeit der konkreten Diagnose. Sowohl während als auch nach der Operation kann es zu Komplikationen kommen. Anhaltende starke Beschwerden nach der Diskotomie werden als »Postdiskotomiesyndrom« (PDS) oder als »Failed Back Syndrom« bezeichnet [Seelig & Nidecker 1989]. In einigen Fällen ist ein weiterer operativer Eingriff erforderlich [Krämer 2006][16]. Ist der Gesundheitszustand des Patienten nach der Operation stabil, kann er in die ambulante Nachsorge entlassen werden. Je nach Schwere des Eingriffs, Beschwerdebild und sozialem Umfeld des Patienten ist die Überführung in eine Rehabilitationsmaßnahme möglich; dieser wird in der Regel vom Sozialdienst des Krankenhauses beantragt.

Wie der bisherigen Prozessbeschreibung entnommen werden kann, gestaltet sich der Behandlungsverlauf bereits ohne eine explizite Betrachtung von Varianz variabel. Variabilität entsteht z.B. im Hinblick auf die zu wählenden Bild gebenden Methoden, die Art der Operation und die Entscheidung für oder gegen eine Rehabilitationsmaßnahme im Anschluss an den stationären Aufenthalt. Eine Einschränkung dieser Variabilität findet jedoch bei einer einrichtungsspezifischen Betrachtung des Behandlungsprozesses statt, da klinische Pfade stets organisationsabhängig entwickelt und eingesetzt werden und sich daher an dem dort üblichen, begrenztem Spektrum an diagnostischen und operativen Verfahren orientieren.

Was als Standard und was als Abweichung vom Standard angesehen wird, muss daher auch innerhalb der jeweiligen Einrichtung definiert werden. Wird z. B. festgelegt, dass bei jedem Patienten mit Verdacht auf lumbalen Bandscheibenvorfall standardmäßig eine Röntgenaufnahme und ein MRT erstellt, aber nur in speziellen Fällen ein CT angefertigt wird, so muss der Durchführung des CTs ein vom Standard abweichendes Ereignis, eine Varianz, vorausgehen. Ebenso stellt die Entscheidung gegen eine Rehabilitationsmaßnahme nur dann eine Abwei-

[16] Vgl. Krämer 2006, S. 288

chung auf Grund einer Varianz dar, wenn ein solcher Beschluss nicht im klinischen Pfad als Standard vorgesehen ist. Im Folgenden werden Beispiele für Varianz angeführt, die grundsätzlich als Abweichung vom Standard betrachtet werden können und nicht von der Definitionshoheit der jeweiligen Einrichtung abhängen. Generell setzt die Deklaration eines Vorfalls als Varianz jedoch voraus, dass Vorgänge zur Berücksichtigung des Vorfalls nicht Bestandteil des klinischen Pfades sind (siehe Kapitel 2.2.2).

2.2.1.2 Varianz 1: Diabetes mellitus

Bei Diabetes mellitus handelt es sich um eine chronische Stoffwechselkrankheit, die sich in abnorm hohen Blutzuckerwerten (Hyperglykämie) äußert. Insgesamt können grob zwei Subtypen von Diabetes mellitus unterschieden werden [Alberti et al. 1998]: Typ-1-Diabetes mellitus hat seine Ursache in einem Mangel an Insulin und tritt meist bereits in jungen Jahren auf. Bei Typ-2-Diabetes handelt es sich hingegen oftmals um eine »Lifestyle«-Erkrankung; Grund ist Insulinresistenz in Folge von anhaltendem Übergewicht, Bewegungsmangel und falscher Ernährung. Diese Form ist häufig im Rahmen des metabolischen Syndroms anzutreffen, bei dem Diabetes mellitus in Kombination mit einer Fettstoffwechselstörung, Adipositas und Bluthochdruck (Hypertonie) die Gefahr für das Auftreten koronarer Herzerkrankungen stark erhöht. Gemäß einer kürzlich durchgeführten Studie wird die Verbreitung von Typ-2-Diabetes mellitus in den kommenden Jahren enorm zunehmen und bis zum Jahr 2030 Rekordwerte annehmen [Wild et al. 2004]. Angesichts dieser Entwicklung ist die Berücksichtigung der Vorerkrankung Diabetes mellitus bei der Planung der Patientenbehandlung gemäß klinischem Pfad wesentlich.

Im Hinblick auf die Behandlung von Wirbelsäulenerkrankungen und speziell in Anbetracht von operativen Eingriffen stellt Diabetes mellitus einen Risikofaktor dar, der das Therapieergebnis negativ beeinflussen kann [Simpson et al. 1993, Wimmer et al. 1998]. Um auf die speziellen Bedürfnisse von Diabetes Patienten angemessen einzugehen, müssen im Rahmen der Diagnostik zusätzliche Untersuchungen durchgeführt werden. Zu den durch Diabetes mellitus bedingten präoperativen Untersuchungen zählen u. a. [Doherty 2006][17]:

- Regelmäßige Messung und Überwachung des Blutzuckerwertes
- Durchführung einer kompletten Harnanalyse zur rechtzeitigen Diagnose und Behandlung von Harnwegsinfektionen und Nierenleiden
- Erstellung eines EKGs (Elektro-Kardiogramms) zum Erkennen einer Herzinsuffizienz
- Anfertigung einer Röntgenaufnahme des Brustkorbs zur Identifikation versteckter Pneumonie oder Lungenödeme
- Gabe von Insulin (während der Operation z. B. durch Infusion)
- Ersetzung der normalen Kost durch Diabetikerkost

Bei der Diagnose von Infektionen oder weiteren Begleiterkrankungen kann es vorkommen, dass der Operationstermin auf einen späteren Zeitpunkt verschoben werden muss. In jedem Fall sollten Diabetes-Patienten mindestens einen Tag vor der Operation stationär aufgenommen werden, um die notwendigen zusätzlichen Untersuchungen durchzuführen; bei schlecht eingestellten Diabetes-Patienten können sogar mehrere Tage notwendig sein. Nach der Ope-

[17] Vgl. Doherty 2006, S. 35

ration muss die Entwicklung des Blutzuckerwertes genau kontrolliert (Messung alle zwei bis vier Stunden), die Wundheilung beobachtet und eventuell auch eine prophylaktische Behandlung mit Antibiotika (Antibiose) durchgeführt werden. Anpassungen an der Verabreichung von Glukose und Insulin orientieren sich an den gemessenen Blutzuckerwerten [Doherty 2006][18].

Soll ein Patient mit Diabetes mellitus auf der Grundlage des klinischen Pfades behandelt werden, so ist es nötig, den Pfad unter Berücksichtigung der Vorerkrankung abzuändern. Dabei kann der Zeitpunkt der Feststellung der Erkrankung variieren und die Diagnose nimmt Einfluss auf alle Phasen des Behandlungsverlaufs (siehe Abbildung 7). Da es sich bei Diabetes mellitus um eine häufig vorkommende Krankheit mit zunehmender Prävalenz handelt, ist es möglich, die zusätzlichen Maßnahmen bereits während der Erstellung der Prozessdefinition des klinischen Pfades zu berücksichtigen. Dabei sollte jedoch beachtet werden, dass diese Vorerkrankung nicht nur bei der Behandlung von Wirbelsäulenerkrankungen, sondern auch im Rahmen anderer Diagnosen, Therapien und damit unabhängig von spezifischen klinischen Pfaden von Bedeutung ist.

2.2.1.3 Varianz 2: Thromboembolische Komplikation

Eine »Thromboembolie« entsteht, wenn körpereigene oder körperfremde Stoffe, wie z.B. ein geronnener Blutklumpen, in die Blutbahn geraten und sich im Gefäßbaum verhaften. Postoperative Thromben können sich in Folge einer lokalen Gefäßwandschädigung bei der Operation ergeben [Seifried & Heinrich 2000][19]. Die venöse Thromboembolie umfasst die Krankheitsbilder »tiefe Bein- oder Beckenvenenthrombose« und »Lungenembolie«. Thromboembolische Komplikationen zählen zu den häufigen Komplikationen nach operativen Eingriffen und haben außerdem einen hohen Stellenwert, da sie zu schwerwiegenden Folgen und sogar zum Tod von Patienten führen können [Kearon 2003, White 2003]. Dieser Umstand steigert ihre Bedeutung als Varianz in Zusammenhang mit einem klinischen Pfad zur Behandlung von Wirbelsäulenerkrankungen.

Die Komplikation tritt meist während des ersten oder zweiten postoperativen Tages auf [Wenninger 2004][20]. Bei vielen Patienten ist die Diagnose der Venenthrombose oftmals erschwert, da sich die charakteristischen Symptome wie Schmerzen und Gefühlsstörungen mit den postoperativen klinischen Befunden überschneiden. Zeigen sich nach der Bandscheibenoperation jedoch Anzeichen für das Vorliegen einer thromboembolische Komplikation, müssen im klinischen Pfad des Patienten sofortige Maßnahmen zur Diagnostik und Therapie eingeschlossen werden. Gemäß medizinischer S2-Leitlinie »Diagnostik und Therapie der Bein- und Beckenvenenthrombose und Lungenembolie« der AWMF orientiert sich die Reihenfolge der zusätzlichen Aktivitäten an der Höhe der klinischen Wahrscheinlichkeit des Vorliegens der Komplikation [AWMF 2005]. Bei einer geringen Wahrscheinlichkeit wird zunächst zur Durchführung eines D-Dimer Tests geraten; bei sehr hoher Wahrscheinlichkeit oder positivem D-Dimer Test sollte als nächstes eine Kompressionssonographie durchgeführt werden. Bei nicht eindeutigem Ergebnis kann zur definitiven Abklärung die Phlebographie eingesetzt werden. Bei älte-

[18] Vgl. Doherty 2006, S. 37
[19] Vgl. Seifried & Heinrich 2000, S. 1
[20] Vgl. Wenniger 2005, S. 90

ren Patienten sollte zusätzlich auf das Vorliegen eines Tumors geprüft werden, der ebenfalls Ursache für eine Beinvenenthrombose sein kann. Ist die Venenthrombose gesichert, ist im Anschluss sofort eine Therapie durch Antikogulation (Verabreichung gerinnungshemmender Mittel) durchzuführen, um das Risiko einer Lungenembolie zu minimieren [Hull et al. 1997]. Bei einem Verdacht auf Lungenembolie sind weitere diagnostische und therapeutische Maßnahmen einzuplanen und durchzuführen [AWMF 2005]. Im Fall einer starken Beeinträchtigung des Patienten durch die Lungenembolie kann ergänzend zur Antikogulation eine Therapie durch Thrombolyse oder eine Embolektomie eingeleitet werden [Seifried & Heinrich 2000][21]. Das Vorliegen einer Thromboembolie kann eine deutliche Erhöhung der stationären Verweildauer der Patienten bewirken [Wenniger 2004][22].

Das Beispiel »thromboembolische Komplikation« zeigt, dass die Folgen einer Varianz nicht immer komplett abgeschätzt werden können, sondern sich die Varianz schrittweise entwickeln kann. Dementsprechend muss auch die Anpassung des klinischen Pfades dem aktuellen Erkenntnisstand folgend in einem iterativen Prozess vorgenommen werden. Wie bei Diabetes mellitus gilt auch bei dieser Varianz, dass ihr Auftreten nicht an die Behandlung lumbaler Bandscheibenvorfälle gebunden ist, sondern auch in Zusammenhang mit anderen Behandlungsprozessen vorkommt.

2.2.1.4 Varianz 3: Duraläsion

Bei der Duraläsion handelt es sich um eine relativ häufig auftretende Komplikation der Wirbelsäulenchirurgie. Die Dura repräsentiert die harte Haut des Wirbelkanals, die während einer Operation zufällig verletzt werden kann. Faktoren, die Duraläsionen begünstigen, sind bestimmte anatomische Gegebenheiten, bereits erfolgte Operationen an der Wirbelsäule, Unerfahrenheit des Operateurs sowie mangelnde instrumentelle Ausstattung [Kurt 2006][23].

Das Anzeichen einer Duraläsion ist das Austreten von Rückenmarksflüssigkeit aus der Wunde. Die Versorgung richtet sich nach der Größe der Verletzung. Während bei einem geringfügigen Riss eine selbständige Ausheilung möglich ist, werden größere Duraläsionen intraoperativ entweder mit einer Fettlappenplastik abgedeckt oder durch eine Naht mit zusätzlicher Applikation von Fibrinkleber versorgt [Kurt 2006[24] 38, Kreuzer 2006[25]]. Folgen einer Duraverletzung sind postoperativ oftmals Kopfschmerzen. Kommt es nach der Operation weiterhin zum Verlust von Rückenmarksflüssigkeit in Folge der Duraläsion, kann sich eine persistierende Liquorfistel bilden. In diesen Fällen können die Anlage einer lumbalen Liquordrainage und eine temporäre fraktionierte Liquorableitung zur Abheilung führen. Alternativ muss ein weiterer operativer Eingriff zum endgültigen Fistelverschluss vorgenommen werden [Kreuzer 2006][26]. Während die frühzeitige Mobilisation von Patienten nach einer Bandscheibenoperation einen hohen Stellenwert hat, wird Patienten mit Duraläsionen eine längere Immobilisationsphase

[21] Vgl. Seifried & Heinrich 2000, S. 85
[22] Vgl. Wenniger 2005, S. 92
[23] Vgl. Kurt 2006, S. 37
[24] Vgl. Kurt 2006, S. 38
[25] Vgl. Kreuzer 2006, S. 49
[26] Vgl. Kreuzer 2006, S. 57-58

verordnet. Die Folgen einer Duraläsion für den klinischen Pfad des Patienten können daher in einer Verlängerung des stationären Aufenthalts resultieren. Im Gegensatz zu den bisher betrachteten Varianzen steht diese Komplikation eng in Zusammenhang mit Operationen an der Wirbelsäule und hat daher keine Bedeutung für klinische Pfade zur Behandlung anderer Diagnosen.

2.2.1.5 Varianz 4: Unvollständigkeit der Befunde

Bei der stationären Aufnahme des Patienten im Krankenhaus kann es vorkommen, dass notwendige Testresultate und Vorbefunde nicht vorhanden sind. Grund dafür ist entweder, dass die Operation im Rahmen der ambulanten Versorgung nicht gründlich geplant wurde und entsprechende Tests nicht durchgeführt wurden, dass aufgrund mangelhafter Ausstattung der niedergelassenen Ärzte die notwendigen Untersuchungen nicht möglich waren oder dass der Patient die erforderlichen Unterlagen nicht vollständig zum Termin mitgebracht hat. Können die Befunde nicht am Tag der Aufnahme nachgeliefert werden, kann dieser Umstand dazu führen, dass die Operation verschoben werden muss, bis die Befunde nachgereicht wurden oder dass die Untersuchungen am Aufnahmetag in der Klinik (zum wiederholten Mal) ausgeführt werden müssen. Je nachdem, um welche Befunde es sich handelt, kann sich sogar die Verweildauer des Patienten erhöhen. Die zusätzlichen Untersuchungen und ggf. der zusätzliche Behandlungstag müssen entsprechend in den klinischen Pfad des Patienten aufgenommen werden. Untersuchungen, die aufgrund der Ausstattung häufig nur im Krankenhaus durchgeführt werden können, sind Myelographie und Discographie. Hier ist zunächst die Injektion eines Kontrastmittels notwendig, bevor Röntgen- oder CT-Untersuchungen durchgeführt werden. Die Myelographie wird genutzt, um z. B. rezidive Bandscheibenvorfälle von postoperativen Vernarbungen zu unterscheiden. Die Discographie ermöglicht die Unterscheidung zwischen Bandscheiben-bedingten Schmerzen und dem Facetten-Syndrom. Im Extremfall muss über die Notwendigkeit der Operation nach Erstellung der fehlenden Befunde neu entschieden werden.

Die Varianz unterscheidet sich von den vorher geschilderten dadurch, dass nicht nur medizinische sondern auch organisatorische Gründe vorliegen können, die zur Abweichung vom ursprünglich im klinischen Pfad vorgesehenen Behandlungsverlauf führen.

2.2.2 Begriffsklärung und Definition

Aus dem Beispiel der Behandlung von Wirbelsäulenerkrankungen wird die Vielfalt von Varianz deutlich und auch die Schwierigkeit zu einer einheitlichen Definition zu gelangen. Einerseits lässt sich Varianz einrichtungsunabhängig mit der Diagnose von Nebenerkrankungen und der Feststellung von Komplikationen in Verbindung bringen und andererseits resultiert sie aus einrichtungsabhängigen Definitionen von Abweichungen von Standards. In einigen Fällen können die Konsequenzen einer Varianz für die patientenbezogene Prozessinstanz des klinischen Pfades klar definiert werden, z.B. auf Basis von krankenhausinternen Richtlinien und Praktiken oder medizinischen Leitlinien. In anderen Situationen ist die Entwicklung der Varianz im Voraus nicht abschätzbar und die Anpassung des Pfades erfolgt schrittweise analog zum Aufschlussgewinn über das Ausmaß der Varianz. Teilweise ist die Variabilität jedoch auch so groß, dass es nicht möglich ist, ein Standardverfahren für den Umgang mit der Varianz zu empfehlen, sondern die notwendigen Entscheidungen auf Fallbasis getroffen werden müssen. Für das Auftreten einer Varianz können sowohl Änderungen in den medizinischen als auch

organisatorischen Rahmenbedingungen, die der Prozessdefinition des klinischen Pfades zugrunde liegen, verantwortlich sein. Insgesamt lässt sich feststellen, dass viele der beschriebenen Varianzen nicht nur in Zusammenhang mit der Behandlung von Wirbelsäulenerkrankungen wichtig sind, sondern auch bei anderen klinischen Versorgungsszenarien Relevanz haben.

An Hand einer Analyse von Literaturquellen zum Thema Varianz von klinischen Pfaden soll das Phänomen in diesem Kapitel nun präzisiert werden. Wie bereits in Kapitel 2.1 erläutert, repräsentieren klinische Pfade einen einrichtungsspezifischen Prozess zur Behandlung von Patienten ausgehend von einer bestimmten Diagnose oder einem spezifischem Therapieverfahren. Implizit enthalten in der Definition klinischer Pfade ist also die Forderung, die medizinische Patientenversorgung stärker zu standardisieren und Abweichungen vom Standard zu reduzieren [Panella et al. 2003]. Merriam-Webster's Wörterbuch zur Folge wird Varianz als

> »the state of being variable or variant« und »the state of being in disagreement« definiert [Merriam-Webster 2003].

In der Literatur wird Varianz allgemein und weitgehend unabhängig von konkreten Elementen des Metamodells zur Spezifikation von Prozessdefinitionen, wie z.B. Aktivitäten, Kontroll- oder Datenflüssen, behandelt. Die folgende Tabelle gibt einen Überblick über Definitionen in Zusammenhang mit Varianz aus Sicht von Domänenexperten.

Tabelle 3 Überblick über Definitionen für Pfadvarianz

Literaturreferenz	**Definition**
Atwal & Caldwell 2002	»...any deviation from the proposed standard of care listed in the pathway«
Barth Frink & Strassner 1996	»... deviation of actual outcomes from a predetermined norm, standard, rate, goal, threshold, or expected outcome«
Bowers 1995	»... a deviation from the patient care activities outlined on the critical path... which may alter the anticipated discharge date, the expected cost, or the expected outcomes«
Bryan et al. 2002	»... deviations from the pathway standards. ... As the pathway evolves and is modified, fewer system- and community-derived anomalies should be detected«
Cheah 1998	»... the unexpected events that occur during patient care - events that are different from what is predicted on the clinical pathways. ... using clinical pathways ... can reduce avoidable variation in the clinical process«
Currie & Harvey 2000	»... any deviation from the pathway ...«
de Luc 2000	»Any deviation from the care plan... unexpected events ... which are different from those predicted in the pathway«
Fox et al. 2003	»... any deviation from the expected care identified on the care pathway, as without the ability for health care professionals to vary treatment programmes according to patient needs, care cannot be individualised.«
Jones 2000	»If the patient or care provider deviates from the care pathway«
Lowe 1998	»Pathways... allow for individualization through the use of variation. Variation allows for documentation of a change from the pathway to suit the individual patient or a change in the

	situation«
Luttman et al. 1995	»...any activity (treatment, procedure, test, or clinical outcome) that does not occur at all, or does not occur in the scheduled time period«
Walsh 1997	»... the discrepancy between expected and actual events«, including »comorbidity and personal psychosocial factors«
Wigfield & Boon 1996	»... differences between what is expected to happen and what does happen«
Zander 2002	»... the difference between the item stated within the time period stated and the actual event«

Den meisten Definitionen zur Folge entspricht Varianz irgendeiner Form von Abweichung vom klinischen Pfad; einige Autoren spezifizieren die Pfadelemente, auf die sich die Abweichung bezieht exakter, wie z.B. Barth Frink & Strassner [Barth Frink & Strassner 1996] und Luttman et al. [Luttman et al. 1995]. Um die Fülle an Varianzdokumentation einzuschränken, werden in der Literatur teilweise nur solche Abweichungen als zu dokumentierende Varianz deklariert, die sich auf das erwartete Entlassdatum, die Kosten oder die Qualität der Behandlung auswirken [Bowers 1995].

Auch im Hinblick auf die Akzeptanz und Notwendigkeit von Varianz gibt es unter den Domänenexperten Meinungsverschiedenheiten. Bryan et al. setzt Varianz von klinischen Pfaden gleich mit mangelhaften Behandlungsergebnissen und erklärt die Elimination sämtlicher Abweichungen vom Standard zum zentralen Ziel klinischer Pfade [Bryan et al. 2002]. Hill betont jedoch, dass eine Varianz vom klinischen Pfad nicht gezwungenermaßen negative, sondern auch positive Auswirkungen auf den Behandlungsverlauf haben kann; die schnelle Regeneration des Patienten nach einer Operation kann z. B. zum Wegfall ursprünglich geplanter Aktivitäten und zu einer Verminderung der Verweildauer führen [Hill 2001]. Eine Reihe von Autoren bewerten Varianz sogar als unverzichtbaren Bestandteil der medizinischen Versorgung, da nur durch das Zulassen von Varianz die Möglichkeit zur Berücksichtigung der individuellen Patientensituation gegeben ist [Campbell et al. 1998, Merritt et al. 1999, Fox et al. 2003]. Diese Ansicht gründet auf der Feststellung, dass sich die Patientenversorgung nie vollständig standardisieren lässt und es immer Abweichungen vom klinischen Pfad geben wird. Zander bewertet die Bedeutung von Varianz von der Ergebnisqualität insgesamt höher als die Varianz im Hinblick auf durchzuführende Aktivitäten oder Interventionen, da sie einen Hinweis auf die Notwendigkeit der Individualisierung klinischer Pfade liefert [Zander 2002].

Allen Definitionen von Varianz von klinischen Pfaden gemeinsam ist das Konzept der Abweichung von einem festgelegten Standard. In Anlehnung an die Ausführungen in Kapitel 2.1.4.2 können der klinische Pfad und dementsprechend auch Instanzen vom klinischen Pfad nur solche Elemente umfassen, die im Wf-Metamodell spezifiziert sind. Mit Blick auf die Modellierung unter Nutzung von WfMS kann Varianz von Prozessen informal definiert werden als

> »Diskrepanz zwischen dem geplanten und dem tatsächlichen Prozessablauf im Hinblick auf den funktionalen, verhaltensorientierten, informationsbezogenen, organisatorischen und/oder operationalen Aspekt der Prozessdefinition und bezogen auf eine konkrete Abweichungsursache.«

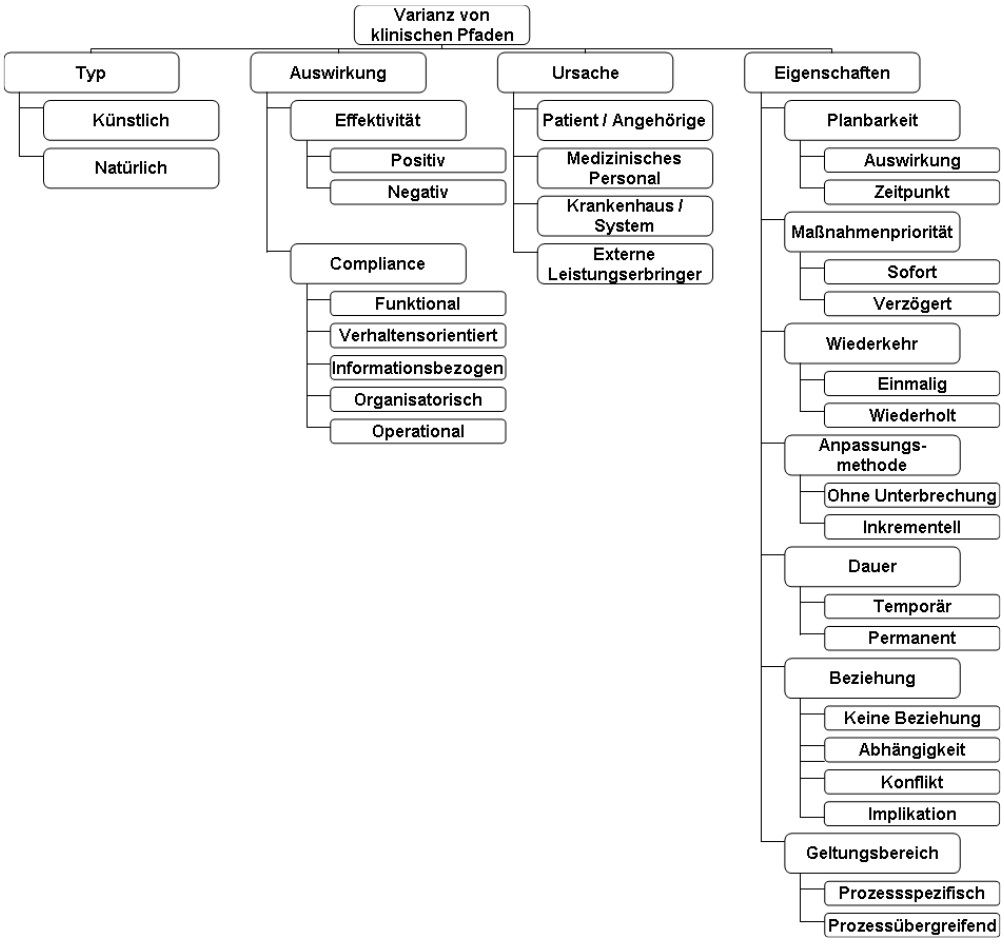
Alle Vorgänge, die aufgrund einer Varianz zu Diskrepanzen zwischen dem tatsächlichen und dem geplanten Prozessablauf führen, werden im Folgenden auch als »Abweichungsmaßnahmen« bezeichnet. Werden diese Maßnahmen in die Prozessdefinition des Pfades integriert, müssen Möglichkeiten bereitstellen, sie klar als Abweichung vom eigentlichen Standard zu

kennzeichnen. Werden Abweichungsmaßnahmen in Zusammenhang mit Varianz aus klinischen Pfaden ausgeschlossen, müssen WfMS Methoden anbieten, durch die Abweichungsmaßnahmen als konkrete Änderungen an Prozessdefinitionen und –instanzen umgesetzt werden.

2.2.3 Anforderungen an das Lösungskonzept

Da unterschiedliche Arten von Varianz unterschiedliche Anforderungen an die Flexibilisierung von WfMS stellen, wird in diesem Kapitel eine Taxonomie zur Klassifikation von Varianz von klinischen Pfaden aufgestellt. Die Entwicklung der Taxonomie basiert auf einer umfangreichen Literaturrecherche und auf Erfahrungen in Projekten zur Erhebung klinischer Prozesse und Abweichungen [Claasbrummel et al. 2007, Reuter & Neuhaus 2008]. Die nachfolgende Abbildung gibt einen Überblick über die einzelnen Klassifikationsmerkmale von Varianz und ihren möglichen Ausprägungen. Ausgehend von diesen Merkmalen werden Anforderungen identifiziert, die ein Lösungskonzept zur prozessorientierten Umsetzung klinischer Pfade erfüllen müssen, um mit dem Auftreten von Varianz adäquat umzugehen.

Abbildung 8 Taxonomie der Klassifikationsmerkmale für Varianz von klinischen Pfaden



2.2.3.1 Typen von Varianz

Um den Konflikt zwischen Elimination von Varianz in der medizinischen Behandlung durch die Einführung klinischer Pfade und Individualisierung der Patientenversorgung auflösen zu können, treffen Bevan & Cawley eine generelle Unterscheidung zwischen natürlicher und künstlicher Varianz [Bevan & Cawley 2006]. Natürliche Varianz wird als unvermeidbarer Bestandteil des Gesundheitssystems betrachtet; sie entsteht in Folge sozioökonomischer und demographischer Unterschiede in der Patientenpopulation, sowie aufgrund von Variationen im Hinblick auf Symptome, Vorerkrankungen und Komorbidität. Den Autoren zufolge besteht die Notwendigkeit, Wege und Methoden zum Umgang mit natürlicher Varianz von klinischen Pfaden zu finden, um medizinisches und pflegerisches Personal bei der Anpassung klinischer Pfade an individuelle Behandlungskontexte zu unterstützen. Im Gegensatz dazu resultiert künstliche Varianz aus persönlichen Präferenzen oder Prioritäten des behandelnden Personals oder aus Mängeln bei der Abstimmung und Koordination administrativer, ärztlicher und pflegerischer Abläufe aufeinander. Mit künstlicher Varianz wird daher genau das Problemfeld umrissen, das die Einführung des prozessorientierten Paradigmas in Unternehmen befördert (siehe Kapitel 2.1.4.1). Während also natürliche Varianz als elementarer Bestandteil der Patientenversorgung charakterisiert wird, ist eine Elimination der künstlichen Varianz auf Basis klinischer Pfade anzustreben.

Anforderung: Das Lösungskonzept muss grundsätzlich ein Verfahren bereitstellen, dass es für Fachanwender möglich macht, klinische Pfade unter Eliminierung unerwünschter künstlicher Varianz zu modellieren. Um natürliche Varianz zu unterstützen, müssen Fachexperten allerdings auch die Option haben, den klinischen Pfad variabel zu gestalten. Für das medizinische Personal muss der Unterschied zwischen Standard und Varianz sofort ersichtlich sein.

2.2.3.2 Auswirkungen von Varianz

Varianz kann auf die Effektivität der Gesundheitsversorgung und/oder auf die Compliance der tatsächlichen Behandlung zum klinischen Pfad Einfluss nehmen [Luttman 2000]. Effektivität ist ein Begriff, der in engem Zusammenhang mit den Geschäftszielen einer Einrichtung steht und dem daher eher subjektiv betrachtet Inhalte zugeordnet werden können; z. B. kann sich Effektivität auf definierte finanzielle, klinische oder qualitative Aspekte oder auch den Grad an Patientenzufriedenheit beziehen. Wie bereits in Kapitel 2.2.2 erläutert, kann sich Varianz entweder positiv oder negativ auf die Effektivität auswirken. Die Compliance hingegen misst objektiv die Einhaltung des klinischen Pfades bei der Behandlung von Patienten. Während bei Patienten der Gruppe »Pfad-Durchläufer« die Konformität mit dem klinischen Pfad gewährleistet ist, gibt es bei »Pfad-Abweichlern« eine Varianz von der Compliance. Im Extremfall wird der klinische Pfad bei »Pfad-Abrechern« gar nicht mehr befolgt. Während Abweichungen von der Compliance mit der Varianzdokumentation zu belegen sind, kann auf Varianz im Hinblick auf die Effektivität nur durch eingehende Analyse der Dokumentation und der Auswirkungen auf den weiteren Behandlungsfortschritt geschlossen werden.

Anforderung: Um Abweichungen vom klinischen Pfad protokollieren zu können, muss das Lösungskonzept die Mittel bereitstellen, durch die solche Abweichungen von Seiten des Systems automatisch festgestellt werden können. Nur auf dieser Basis lassen sich von der Compliance bzw. Non-Compliance zum klinischen Pfad wertvolle Rückschlüsse auf die damit verbundene Effektivität der medizinischen Behandlung ziehen.

2.2.3.3 Ursachen für Varianz

Betrachtet man die Gründe für Varianz von klinischen Pfaden, so können zwei Klassen gegeneinander abgegrenzt werden [Bevan & Cawley 2006]. Die eine Klasse von Varianz ist zurückzuführen auf allgemein bekannte Ursachen (»common cause«), mit denen während der Patientenbehandlung zu rechnen ist. Beispiele sind bekannte Komplikationen oder häufig auftretende Vorerkrankungen und Risikofaktoren. Die zweite Klasse von Varianz hat spezifische Gründe, die stark individuell geprägt sind und deren Auftreten häufig nicht planbar ist, z.B. allergische Reaktion auf Kontrastmittel. Von Zander stammt ein vierstufiges Klassifikationsschema, das bei den meisten Implementierungen klinischer Pfade die Grundlage für die Erfassung der Varianzdokumentation bildet [Zander 1991]:

- Patient/Familie: Die Abweichung steht direkt in Zusammenhang mit dem Patient und/oder seiner Familie (z. B. Auftreten einer Infektion, Feststellung einer Komorbidität, keine ausreichende Mobilisierung, mangelnde Mitarbeit des Patienten, Beeinflussung durch Angehörige).
- Medizinisches Personal bzw. stationäre Planung: Die Ursache für die Varianz ist beim medizinischen Personal auf Station zu suchen (z. B. Fehlen von Befunden und Testresultaten, Verzögerungen durch andere Untersuchungen, Urlaub oder Feiertage, interne Probleme bei der Entlassungsplanung).
- Krankenhaus/System: Die Abweichung ist zurückzuführen auf Gründe, die generell in Zusammenhang mit der Krankenhaus- und Systemorganisation stehen (z. B. Verzögerung des Patiententransportes, Verzögerung in der OP-Planung, Verzögerung oder Absage eines Konsils, Ausfall eines medizinischen Systems).
- Externe Leistungserbringer: Die Gründe für die Varianz liegen nicht innerhalb des Krankenhauses oder beim Patienten, sondern beziehen sich auf die Versorgung durch ambulante Leistungserbringer bzw. Rehabilitationseinrichtungen (z. B. mangelnde Verfügbarkeit von Rehabilitations- oder Pflegeplätzen, fehlende häusliche Krankenpflege, Facharztmangel in der Region, Verzögerung des externen Transportes).

Neben dem Schema von Zander gibt es weitere Klassifikationsansätze für die Ursachen von Varianz von klinischen Pfaden. An dieser Stelle sei auf die entsprechenden Literaturquellen verwiesen [Robinson et al. 1992, Mikulaninec 1992, Hoffman 1993].

Gründe für natürliche Varianz beziehen sich im Fall von Krankenhäusern in erster Linie auf die Kategorie »Patient / Familie«, da diese weitgehend außerhalb des Einflussbereiches des Klinikpersonals und des Managements liegen. Die spezifische Behandlungssituation des Patienten und seine individuellen Entscheidungen oder die seiner Angehörigen müssen unabhängig von dem im klinischen Pfad geplanten Verlauf weiterhin berücksichtigt werden, auch wenn dies zu Anpassungen an der Behandlungsplanung führt. Weitere Gründe, auf die das Krankenhaus nur bedingt Einfluss nehmen kann, sind in der Kategorie »Externe Leistungserbringer« zu verorten. Obwohl es im Gesundheitswesen einen Trend zur stärkeren einrichtungsübergreifenden Kooperation und Koordination der Leistungen kommt und die Zusammenarbeit sogar mehr und mehr vertraglich geregelt wird (Verträge zur integrierten Versorgung [DKG 2004]), umfassen klinische Pfade die Anteile Externer an der Patientenbehandlung bisher nicht. Um auch die intersektorale Planung besser steuern zu können, gibt es allerdings Bestrebungen, das Konzept klinischer Pfade über die Klinikgrenzen hinaus auch auf andere Leistungserbringer auszudehnen. Ein Beispiel für eine Initiative in dieser Richtung ist das von der Fachhochschule Dortmund in Zusammenarbeit mit der Knappschaft Bahn See (KBS) durchgeführte DiPP Projekt (Digitale Pfade im Gesundheitsnetz Prosper) [Eckenbach & Böckmann 2008, Böckmann 2010]. Das Auftreten von Varianz der Kategorien »Medizinisches Personal«

und »Krankenhaus / System« hingegen ist allein in der Verantwortung des Krankenhausmanagements zu sehen; hierbei handelt es sich also um künstliche Varianz, die weitgehend durch die Definition und den Einsatz klinischer Pfade eliminiert werden soll.

Anforderung: Das Lösungskonzept zur Umsetzung klinischer Pfade muss so ausgerichtet sein, dass es Abweichungen vom Standard prinzipiell unabhängig von der konkreten Ursache unterstützt. Auch wenn eine Varianz auf dieselbe Ursache zurückzuführen ist, muss dennoch die Möglichkeit bestehen, sie in Abhängigkeit des aktuellen Behandlungsfalles variabel zu gestalten.

2.2.3.4 Eigenschaften von Varianz

Während der Recherche zum Thema Varianz von klinischen Pfaden konnten insgesamt acht Eigenschaften von Varianz identifiziert werden, die die Eignung und Wahl potentieller Abweichungsmaßnahmen determinieren.

2.2.3.4.1 Planbarkeit

Das Kriterium der Planbarkeit ist bereits bei der Umsetzung von Prozessstandards durch WfMS von entscheidender Bedeutung. So klassifiziert Georgakopolous et al. die in einem Workflow abgebildeten Prozesse gemäß ihrer Vorhersagbarkeit [Georgakopoulos et al. 1995] und Deiters et al. wenden das Kriterium der Planbarkeit auf kollaborative Workflows an [Deiters et al. 1996]. In Bezug auf Varianz drückt Planbarkeit aus, ob es möglich ist, Abweichungen von der Effektivität und insbesondere von der Compliance zum klinischen Pfad im Voraus - also zur Modellierungszeit vor Eintreten der Varianz zur Laufzeit - zu bestimmen. Entsprechende Ansätze zur Flexibilisierung der Prozessausführung werden von Joeris auch nach a-priori und a-posteriori Flexibilität klassifiziert [Joeris 1999]. Generell müssen zwei Aspekte differenziert betrachtet werden: Auswirkung und Beobachtungszeitpunkt. Bei Vorerkrankungen wie Diabetes mellitus oder Komplikationen wie Thromboembolie sind notwendige Modifikationen am klinischen Pfad weitgehend im Voraus bekannt (a priori). Im Gegensatz dazu treten viele Probleme während der Operation und der Anästhesie unerwartet auf und ihr Effekt hängt stark vom Patienten und dem jeweiligen Problem ab; daher ist eine Planung der Aktivitäten im Voraus kaum möglich und stattdessen eine schnelle ad hoc Reaktion nach Eintritt der Varianz gefordert (a posteriori). Der zweite Aspekt, der die Planbarkeit einer Varianz beeinflusst, ist der Zeitpunkt bzw. der Zeitraum, wann eine Varianz auftritt und beobachtet werden kann. Wir unterscheiden die folgenden Fälle:

- Effekt und Beobachtungszeitpunkt sind beide unbekannt; eine Vorausplanung der Varianz ist daher nicht möglich. Beispiele sind eine allergische Reaktion auf ein Medikament oder ein plötzlich auftretender koronarer Herzinfarkt.
- Der Effekt einer Varianz ist planbar, ihr Beobachtungszeitpunkt jedoch variabel. Dies trifft z. B. auf viele Vorerkrankungen zu. Diese können entweder bereits zum Aufnahmezeitpunkt bekannt sein oder erst während der stationären Diagnostik festgestellt werden. Erkrankungen wie Infektionen können sogar vor oder nach einer Operation auftreten.
- Der Beobachtungszeitpunkt ist bekannt, der Effekt einer Varianz kann jedoch nicht exakt vorhergesagt werden. Diese Varianz tritt z. B. ein, wenn ein Patient bestimmte Untersuchungsmethoden ablehnt.

– Sowohl Effekt als auch Beobachtungszeitpunkt der Varianz sind im Voraus bestimmbar. Z. B. werden bei einem Patienten ab dem Überschreiten eines definierten Lebensjahres grundsätzlich zusätzliche Untersuchungen, wie EKG, durchgeführt.

Der Detaillierungsgrad, in dem die Auswirkungen einer Varianz im Voraus geplant werden, kann je nach Bedarf unterschiedlich sein. Einige Arten von Varianz, wie z. B. das intraoperative Versorgen einer Duraläsion erfordern nur leichtes Abrücken vom klinischen Pfad; detailreiche Schilderungen zu Vorgehensweisen zur Anpassung des Pfades sind hier nicht erforderlich. Die Folgen anderer Typen von Varianz, wie z. B. der Umgang mit einer Thromboembolie sind zu variabel, um ein Standardvorgehen aufzustellen. Wieder andere Varianz, wie z. B. die Berücksichtigung einer Vorerkrankung wie Diabetes mellitus, lässt sich sehr gut beschreiben.

Anforderung: Das Lösungskonzept muss Methoden bereitstellen, die den Aufwand der Fachanwender beim Umgang mit Varianz in Abhängigkeit der Planbarkeit der jeweiligen Abweichungsmaßnahmen möglichst minimal halten. Gleichzeitig ergibt sich auch die Anforderung, Konflikte bei der kombinierten Anwendung unterschiedlicher Methoden zu verhindern.

2.2.3.4.2 Maßnahmenpriorität

Das Ergreifen von Maßnahmen in Folge des Auftretens einer Varianz vom klinischen Pfad kann unterschiedlich priorisiert werden. Varianz, die starke negative Auswirkungen auf die Effektivität der Patientenbehandlung hat, erhält üblicherweise eine hohe Priorität; z. B. wenn durch eine verzögerte Reaktion auf die Varianz der Gesundheitszustand des Patienten beeinträchtigt oder sogar dessen Leben bedroht ist. In solchen Fällen müssen die Umplanung der Behandlung und damit verbundene notwendige Abweichungsmaßnahmen frühzeitig erfolgen (»sofort«). In anderen Situationen kann eine verzögerte Reaktion auf das Auftreten einer Varianz leichter hingenommen werden, z. B. wenn bei der stationären Diagnostik festgestellt wird, dass noch eine zusätzliche Untersuchung durchgeführt werden muss, die jedoch die Zeitplanung der Behandlung nicht negativ beeinflusst (»verzögert«).

Anforderung: Das Lösungskonzept muss für Fachanwender Möglichkeiten bereitstellen, sowohl sofort als auch verzögert durch Anpassung der Behandlungsplanung auf eine Varianz zu reagieren.

2.2.3.4.3 Wiederkehr

Einige Klassen von Varianz vom klinischen Pfad treten üblicherweise nur ein einziges Mal im Laufe der Patientenbehandlung auf (»einmalig«). Dazu zählt z. B. die Diagnostizierung von chronischen Vorerkrankungen oder Schwangerschaften. Das bedeutet, dass auch die Anpassung des klinischen Pfades auf Basis der Varianz nur ein einziges Mal erfolgen muss; Ausnahmen bilden Entwicklungen und Komplikationen in Folge der ursprünglichen Varianz. Varianzen, wie z. B. Infektionen oder andere Komplikationen können im Gegensatz dazu wiederholt während der stationären Versorgung vorkommen (»wiederholt«). In solchen Fällen müssen auch entsprechende Änderungen an der Behandlungsplanung und am klinischen Pfad mehrfach durchgeführt werden.

Anforderung: Das Lösungskonzept muss Fachexperten in die Lage versetzen, identische oder ähnliche Abweichungsmaßnahmen bei Bedarf zu unterschiedlichen Zeitpunkten und an unterschiedlichen Stellen des Prozesses zu wiederholen.

2.2.3.4.4 Anpassungsmethode

Im Hinblick auf die Methode, die zur Anpassung der Instanz des klinischen Pfades, zur Anwendung kommt, können zwei wesentliche Strategien unterschieden werden. Bei einigen Varianzen ist es möglich alle Folgen für die weitere Behandlung abzuschätzen und die Änderung an der Prozessinstanz in einer einzigen Operation komplett durchzuführen (»ohne Unterbrechung«). Beispiele für solche Varianzen sind die Diagnose einer Vorerkrankung wie »Diabetes mellitus« oder das Feststellen des Fehlens von notwendigen Befunden. Andere Varianzen wie z.B. »Thromboembolie« können sich auch nach der initialen Feststellung weiterentwickeln, so dass eine einzige Änderung am klinischen Pfad nicht ausreicht, um angemessen auf die Varianz zu reagieren (»inkrementell«). Im Fall der thromboembolischen Komplikation kann z.B. aus einer Beinvenenthrombose eine Lungenembolie hervorgehen. Anpassungen an klinischen Pfaden folgen dementsprechend einer inkrementellen Strategie; d. h. nach jeder Änderung wird mit dem Fortschreiten der Behandlung geprüft, ob weitere Änderungen in Folge der Varianz erforderlich sind.

Anforderung: Das Lösungskonzept muss es möglich machen, die in Folge einer Varianz notwendig gewordenen Abweichungsmaßnahmen komplett mit minimalem Aufwand für Fachexperten oder inkrementell vorzunehmen.

2.2.3.4.5 Dauer

Varianz vom klinischen Pfaden unterscheidet sich weiterhin in Bezug auf die Dauer für die sie Gültigkeit besitzt. Bei Patienten mit Herzschrittmacher ist der Einsatz eines MRTs für die Bild gebende Diagnostik kontraindiziert. Die Varianz hat während der gesamten Patientenbehandlung Gültigkeit (»permanent«) Dementsprechend müssen alle radiologischen Untersuchungen, in denen ein MRT vorgesehen ist, durch eine alternative Methode ersetzt werden. Die prophylaktische Gabe von Antibiotika kann hingegen auf eine spezielle Region des klinischen Pfades beschränkt sein; z. B. Antibiose ab dem Tag der stationären Aufnahme bis zum dritten postoperativen Tag. Die Varianz gilt damit nur für diesen spezifizierten Zeitraum (»temporär«). Im Fall einer permanenten Varianz ist es für das Klinikpersonal nicht zumutbar, dieselbe Änderung an unterschiedlichen Stellen der Prozessinstanz immer wieder durchzuführen.

Anforderung: Das Lösungskonzept muss Mittel bereitstellen, dieselben Abweichungsmaßnahmen gleichzeitig an mehreren Stellen im Prozess vorzunehmen, um den Aufwand für Fachanwender minimal zu halten.

2.2.3.4.6 Beziehung

Die Anpassung eines klinischen Pfades in Konsequenz des Auftretens einer Varianz kann vollkommen unabhängig von anderer Varianz stattfinden (»keine Beziehung«). Je mehr unterschiedliche Varianz in Zusammenhang mit einem klinischen Pfad berücksichtigt wird, desto mehr Beziehungen können sich jedoch zwischen den Varianzen und ihren Auswirkungen auf

den klinischen Pfad ergeben, so dass eine separate Betrachtung unmöglich ist. Ob Varianz zueinander in Beziehung steht, hängt natürlich auch von der Dauer der jeweiligen Varianz ab. Prinzipiell können drei Klassen von Beziehungen unterschieden werden:

- Eine Varianz ist abhängig vom Auftreten einer anderen Varianz. Z. B. ist die Diagnostik von Gestationsdiabetes nur bei Vorliegen einer Schwangerschaft möglich.
- Eine Varianz steht in Konflikt mit einer anderen Varianz oder deren Konsequenzen. Leidet ein Patient z.B. unter einer Kontrastmittelallergie, darf in Folge des Verdachts auf Beinvenenthrombose keine Phlebographie erstellt werden, auch wenn es sich dabei im entsprechenden Krankenhaus um das Diagnosemittel der Wahl handelt.
- Eine Varianz impliziert eine andere Varianz. Z. B. begünstigt Diabetes mellitus das Risiko für Infektionen. Das Vorliegen der Varianz Diabetes mellitus kann also dazu führen, dass eine prophylaktische Antibiose durchgeführt werden muss.

Anforderung: Das Lösungskonzept muss Methoden bereitstellen, um Zusammenhänge zwischen Varianz deutlich zu machen und das Treffen entsprechender Abweichungsmaßnahmen in ihren Kontext zu setzen.

2.2.3.4.7 Geltungsbereich

Der Geltungsbereich einer Varianz kann sich entweder auf genau einen klinischen Pfad (»prozess-spezifisch«) oder auf mehrere klinische Pfade (»prozess-übergreifend«) erstrecken. Ein Beispiel für eine prozess-spezifische Varianz ist die Duraläsion bei der Wirbelsäulenchirurgie; diese Varianz tritt nur in Zusammenhang mit Operationen an der Wirbelsäule auf. Die Häufigkeit prozess-spezifischer Varianz hängt auch von der Menge der für einen Diagnose- und/oder Therapiekomplex vorhandenen klinischen Pfade ab. Gibt es für die Wirbelsäulenchirurgie nur einen klinischen Pfad, muss auch die Duraläsion nur im Kontext dieses Pfades betrachtet werden. Werden stattdessen für unterschiedliche Operationsformen unterschiedliche klinische Pfade entwickelt, so hat auch die Duraläsion einen prozess-übergreifenden Gültigkeitsbereich.

Das Auftreten von Vorerkrankungen, wie Diabetes mellitus, Hypertonie, Adipositas, koronare Herzerkrankungen und deren Komplikationen, ist generell komplett unabhängig vom jeweiligen klinischen Pfad. Der Gültigkeitsbereich dieser Varianzen ist also auch ungeachtet der Menge an klinischen Pfaden als global anzusehen. Dasselbe gilt für allgemeine Risiken bei der operativen Therapie, wie z. B. Harnwegs- und Wundinfektionen, starke Blutungen oder Thrombose. Abhängigkeiten zwischen Varianz mit prozess-übergreifendem Gültigkeitsbereich und klinischen Pfaden können sich jedoch ergeben, wenn die Konsequenzen der Varianz je nach Pfad stark variieren.

Anforderung: In Abhängigkeit des Geltungsbereichs einer Varianz muss es möglich sein, planbare Abweichungsmaßnahmen so zu spezifizieren, dass sie für mehrere klinische Pfade genutzt werden können.

2.3 Zusammenfassung der Anforderungen

Aus den Betrachtungen zu klinischen Pfaden und dem Phänomen der Varianz lässt sich schließen, dass Varianz genau wie klinische Pfade einrichtungsabhängig definiert werden muss. Je nachdem welche diagnostischen und therapeutischen Maßnahmen in den Standardablauf, der durch den klinischen Pfad abgebildet wird, aufgenommen werden, handelt es sich

bei allen davon abweichenden Ereignissen und Vorgängen um Varianz. Weiterhin lässt sich feststellen, dass der klinische Pfad nicht dazu dient, die Möglichkeiten zum individuellen Eingehen auf jeden Patienten willkürlich einzuschränken und die medizinische Versorgung entgegen dem Wohle der Patienten zu vereinheitlichen. Mit klinischen Pfaden soll lediglich das Auftreten von künstlicher Varianz begrenzt werden, für das es keine klinische Ursache gibt und das häufig zu Verzögerung und Ineffizienz bei der Behandlung führt. Abweichungen von klinischen Pfaden sind möglich und im Falle von natürlicher Varianz auch unbedingt erforderlich.

Um klinische Pfade adäquat umzusetzen, müssen WfMS die Definition von Standardprozessen unterstützen, die als Grundlage für die Behandlungsplanung und Automatisierung administrativer Tätigkeiten wie Auftragskommunikation, Ressourcenanforderung, Terminierung und Dokumentation dienen können. Um den Aufwand für Entwicklung und Änderung klinischer Pfade zu reduzieren, müssen sie außerdem Methoden bereitstellen, die es Fachexperten ohne tiefgreifende IT-Kenntnisse möglich machen, selbständig Prozessdefinitionen zu erstellen. Darüber hinaus müssen Lösungsansätze für Prozessmanagement im Krankenhaus jedoch auch flexibel genug sein, um den Fachanwendern den Umgang mit Varianz zu ermöglichen. Demnach sind Abweichungsmaßnahmen notwendig, die geeignet sind, um den unterschiedlichen Anforderungen der vielfältigen Arten von Varianz zu begegnen und den Aufwand für Fachanwender in jedem Fall auf ein Minimum zu begrenzen. So sind manuell durchgeführte Abweichungsmaßnahmen unumgänglich, um auf natürliche Varianz der Kategorien »Patient/Familie« und »Externe Leistungserbringer« zu reagieren, deren Auswirkungen extrem variabel sind oder deren hohe Priorität keine Verzögerungen gestattet. Unterstützt ein WfMS aber auch im Fall aller anderen Klassen von natürlicher Varianz ausschließlich manuelle Anpassungen, ist die Durchführung von Maßnahmen zur Abweichung vom klinischen Pfad für Anwender mit hohem Arbeitsaufwand verbunden. Reduktion von Arbeitsaufwand, Benutzerfreundlichkeit und Bedienbarkeit sind jedoch wesentliche Faktoren, die die Akzeptanz der Anwender im Krankenhaus gegenüber Informationstechnologie beeinflussen. Daher müssen WfMS auch Methoden verfügbar machen, mit denen notwendige Abweichungsmaßnahmen vom Pfad gebündelt werden können, um auch komplexe Anpassungen effizient und standardisiert durchzuführen. Schließlich sind zusätzlich Mischlösungen erforderlich, die es Fachexperten erlauben, im Sinne der Patientinnen und Patienten auch Einfluss auf solche standardisierten Abweichungsmaßnahmen zu nehmen. Die Standardisierung des Umgangs mit Varianz kann nach dem Vorbild der Prozessstandardisierung durch die Einführung klinischer Pfade erfolgen und ist mit einer Reihe von Vorteilen verbunden:

- Reduktion von künstlicher Varianz auch bei Abweichungen vom Standard
- Reduktion des Aufwandes für Abweichungen durch die Selektion eines vordefinierten Maßnahmenbündels
- Minderung des Aufwands für Varianzdokumentation, da der Grund für die Varianz mit der Wahl der definierten Abweichungsmaßnahmen feststeht
- Möglichkeit der Spezifikation von Beziehungen zwischen Varianz und Abweichungsmaßnahmen
- Erleichterung der Varianzanalyse durch die feste Zuordnung von Änderungsmaßnahmen zu Varianz

Vordefinierte Bündel von Änderungsmaßnahmen im Kontext einer oder mehrerer Varianzen werden im Folgenden als »Prozessvariante« bezeichnet. Ausgehend von der in dieser Arbeit angestellten Analyse klinischer Pfade und Pfadvarianz können insgesamt die folgenden zentralen Anforderungen definiert werden:

- Entwicklung eines Formalismus, auf dessen Grundlage die Erstellung von klinischen Pfaden als ausführbare Standardprozesse auch für Fachexperten möglich ist
- Ermöglichung der Durchführung manueller Abweichungsmaßnahmen für Fachanwender sowohl vor als auch nach der Instanziierung des Pfades für einen Patienten
- Entwurf und Umsetzung eines Konzeptes zur Spezifikation von Prozessvarianten als vordefinierte Bündel von Abweichungsmaßnahmen, die von Fachexperten ausgewählt und bei Bedarf noch konfiguriert werden können

In Tabelle 4 sind Teilanforderungen definiert, die erfüllt werden müssen, um die genannten zentralen Anforderungen zu realisieren. In Klammern ist jeweils die Zuordnung zu dem entsprechenden Kapitel der Anforderungsanalyse angegeben.

Tabelle 4 Übersicht über die Anforderungen an die Umsetzung klinischer Pfade und den Umgang mit Varianz

Zentrale Anforderung		**Resultierende Teilanforderung**	
1.	Entwicklung und Ausführung klinischer Pfade (siehe Kapitel 2.1.5)	1.1	Klinische Pfade müssen für Fachanwender modellierbar sein
		1.2	Optimierung des Grades an Wiederverwendbarkeit durch formale Methoden zur Verstetigung des Prozesswissens
		1.3	Semantische Abhängigkeiten zwischen Prozessteilen müssen formalisiert und bei der Modellierung berücksichtigt werden
		1.4	Die Modellierung unterschiedlicher Prozessteile kann gleichzeitig und gleichartig erfolgen
		1.5	Die Ausführung klinischer Pfade nach der Modellierung muss mit unterschiedlichen WfMS möglich sein
		1.6	Es besteht die Möglichkeit, klinische Pfade in unterschiedliche Darstellungsformen zu überführen
2.	Unterstützung manueller Abweichungsmaßnahmen	2.1	Manuelle Abweichungsmaßnahmen vom klinischen Pfad müssen für Fachanwender durchführbar sein (siehe Kapitel 2.2.3.1, 2.2.3.4.1, 2.2.3.4.4)
		2.2	Fachanwender und WfMS müssen klar zwischen Standardprozess und vom Standard abweichenden Maßnahmen unterscheiden können (siehe Kapitel 2.2.3.1, 2.2.3.2)
		2.3	Die Wahl der Abweichungsmaßnahmen muss in Abhängigkeit des Behandlungsfalls variieren können (siehe Kapitel 2.2.3.3)
		2.4	Fachexperten müssen sowohl die Möglichkeit haben sofort als auch verzögert auf eine Varianz zu reagieren (siehe Kapitel 2.2.3.4.2)
		2.5	Abweichungsmaßnahmen können zu unterschiedlichen Zeitpunkten und an unterschiedlichen Stellen im Prozess wiederholt werden (siehe Kapitel 2.2.3.4.3)
		2.6	Dieselben Abweichungsmaßnahmen lassen sich gleichzeitig an mehreren Stellen des Prozesses durchführen (siehe Kapitel 2.2.3.4.5)
		2.7	Semantische Abhängigkeiten zwischen Abweichungsmaßnahmen müssen bei ihrer Auswahl berücksichtigt

			werden (siehe Kapitel 2.2.3.4.6)
		2.8	Dieselben Abweichungsmaßnahmen können unabhängig von bestimmten klinischen Pfaden durchgeführt werden (siehe Kapitel 2.2.3.4.7)
		2.9	Die Ausführung des klinschen Pfades muss auch nach der Durchführung manueller Abweichungsmaßnahmen möglich sein (siehe Kapitel 2.1.5)
3.	Unterstützung von Prozessvarianten	3.1	Prozessvarianten als standardisierte Bündel von Abweichungsmaßnahmen müssen für Fachanwender modellierbar sein (siehe Kapitel 2.2.3.4.1, 2.2.3.4.4)
		3.2	Fachanwender und WfMS müssen klar zwischen Standardprozess und Variante unterscheiden können (siehe Kapitel 2.2.3.1, 2.2.3.2)
		3.3	Semantische Abhängigkeiten zwischen einzelnen Abweichungsmaßnahmen einer Prozessvariante müssen berücksichtigt werden (siehe Kapitel 2.2.3.4.6)
		3.4	Fachanwender müssen Prozessvarianten in Abhängigkeit des Behandlungsfalls anpassen können (siehe Kapitel 2.2.3.4.1, 2.2.3.4.4)
		3.5	Prozessvarianten können unabhängig von bestimmten klinischen Pfaden definiert und eingesetzt werden (siehe Kapitel 2.2.3.4.7)
		3.6	Konflikte mit manuellen Abweichungsmaßnahmen werden vermieden (siehe Kapitel 2.2.3.4.1)
		3.7	Die Ausführung des klinischen Pfades muss auch nach der Auswahl einer Prozessvariante möglich sein (siehe Kapitel 2.1.5)

3 Verwandte Arbeiten

In der Literatur wurden die Einschränkungen konventioneller WfMS bei ihrer Anwendung auf Domänen, die einer hohen Dynamik ausgesetzt sind, erkannt und spezielle Lösungen entwickelt. Auffallend ist, dass vor allem klinische Abläufe zur Analyse von Anforderungen an eine flexible Prozessausführung dienen, die sich schnell an geänderte Rahmenbedingungen anpassen lässt und so die individuelle Situation von Patienten bei der Behandlung adäquat berücksichtigt. Dadam et al. bezeichnen klinische Prozesse sogar als »Killer Application« für prozessorientierte Anwendungssysteme [Dadam et al. 2000]. Auf Basis der Erkenntnis, dass das Konzept des WfM nur dann auf das Gesundheitswesen übertragen werden kann, wenn es dessen hohe Flexibilitätsanforderungen erfüllt, wurden in den letzten Jahren verschiedene, miteinander konkurrierende Flexibilisierungsparadigmen entwickelt [Schonenberg et al. 2008]. In diesem Kapitel werden praktische und wissenschaftliche Lösungsansätze vorgestellt und diskutiert, die einen Beitrag zu den in Kapitel 2.3 aufgelisteten Anforderungen leisten können.

3.1 Ansätze zur Entwicklung und Ausführung klinischer Pfade

Die Notwendigkeit einer bereichsübergreifenden, qualitativ hochwertigen und kosteneffizienten Gestaltung der Behandlungsabläufe, hat dazu geführt, dass das Interesse von Krankenhäusern an einer IT-basierten Prozessteuerung in den letzten Jahren enorm zugenommen hat. Dementsprechend ergeben sich auch neue Anforderungen an das zentrale IT-Instrument in Kliniken, das Krankenhausinformationssystem (kurz: KIS), dessen Entwicklung sich bereits in der dritten Generation befindet [Kleemann 2010]. Während Systeme der ersten Generation lediglich der Erfassung und Verwaltung der Patientenstammdaten, der Abrechnung von Leistungen und der ärztlichen Primärdokumentation nach gesetzlichen Vorgaben dienten, erfüllen die aktuellen Systeme ein weit größeres Aufgabenspektrum. Dazu zählen Funktionen des Business Intelligence (BI) und Auswertungen zur strategischen Unternehmenssteuerung ebenso wie die Anbindung an einrichtungsübergreifende elektronische Akten und die Planungs- und Prozessunterstützung durch die Umsetzung klinischer Pfade. In diesem Kapitel sollen praktische Systemlösungen vorgestellt und im Hinblick auf die Erfüllung der in dieser Arbeit definierten Anforderungen untersucht werden.

3.1.1 Klinische Pfade bei der deutschen Rentenversicherung Knappschaft-Bahn-See

Bereits im Jahr 2003 hat die aus der früheren Bundesknappschaft hervorgegangene Institution Knappschaft-Bahn-See (kurz: KBS) begonnen, in einer Einrichtung in Bottrop klinische Pfade einzuführen [Salfeld et al. 2009][27]. Initial wurden in acht unterschiedlichen Fachabteilungen 80 Pfade erstellt und anschließend schrittweise ausgerollt. Zu den positiven Effekten der Pfadeinführung zählen sinkende Verweildauern und Kosten, Verringerung des Leistungskonsums z.B. in den Bereichen Radiologie und Labor sowie die kostenneutrale Bewältigung des Patientenaufkommens bei Zunahme des Schweregrades der Erkrankung. Mittlerweile werden in Bottrop

[27] Siehe Salfeld et al. 2009, S. 56-61

ca. 75% aller Fälle auf Basis von klinischen Pfaden behandelt. Die ursprünglich papierbasierten Versionen wurden ausnahmslos in das Krankenhausinformationssystem iMedOne® übertragen, in das ein speziell für die Unterstützung von klinischen Pfaden entwickeltes Modul »DocPath« integriert wurde. Das erfolgreiche Konzept wurde ausgehend von Bottrop auch auf die anderen Kliniken der KBS angewandt; darüber hinaus wird das IT-System derzeit so erweitert, dass über das einzelne Krankenhaus hinaus sektorübergreifende Pfade unterstützt werden können [Böckmann 2010].

Die Pfadmodellierung wurde bewusst so gestaltet, dass sie auch für das medizinische und administrative Personal möglich ist. Klinische Pfade werden in einzelne Module zerlegt, die den Kategorien des von der Knappschaft erstellten Pfadkonzeptes zugeordnet sind, wie z. B. Labor, Ernährung, Pflegemaßnahmen etc. Diese Module können in einer kalenderartigen Darstellung angeordnet und mit Informationen zu Dauer, Kosten oder Einschlusskriterien versehen werden. Mit den Pfadelementen können Ereignisse des KIS verknüpft werden, wie z. B. Auftragsdokumentation oder Dokumenterstellung in der elektronischen Patientenakte. Die problemlose Ausführbarkeit eines Pfades kann von Seiten des Systems jedoch nicht automatisch gewährleistet werden, sondern liegt in der Verantwortlichkeit der IT-Spezialisten. D. h. es existiert kein Formalismus, der automatisch sicherstellt oder zumindest im Nachhinein erkennt, dass klinische Pfade nach ihrer Erstellung wirklich ausführbar sind. Z. B. wird nicht überprüft, ob ein Datenobjekt, das von einem Pfadelement benötigt wird, zuvor von einem anderen Element erstellt worden ist. Durch die direkte Integration mit dem KIS ist die Ausführung durch andere WfMS und die Validierung durch deren formale Prozessmodellierungssprachen generell nicht möglich.

Semantische Abhängigkeiten zwischen einzelnen Pfadelementen lassen sich zwar durch eine »Vorgängerrelation« einstellen; jedoch auch hier gibt es keine formalen Methoden, die sicherstellen, dass die Verknüpfung der Elemente per Vorgängerrelation fehlerfrei, also z. B. ohne Schleifen stattfinden. Darüber hinaus existiert kein definiertes Verfahren, wie das Konzept um weitere Relationstypen ergänzt werden kann. Durch die enge Bindung ans KIS werden auch keine generell unterschiedlichen Darstellungsmöglichkeiten z. B. in Abhängigkeit der Einrichtung oder Fachgruppe unterstützt; es lassen sich lediglich Sichten definieren, nach denen Pfadelemente ein- oder ausgeblendet werden, je nachdem welcher Benutzergruppe der aktuelle Betrachter angehört. Die nachfolgende Tabelle fasst die Analyseergebnisse hinsichtlich der Anforderungen an die Entwicklung und Ausführung klinischer Pfade zusammen.

Tabelle 5 Ergebnisse der Analyse von iMedOne.DocPath zur Entwicklung und Ausführung klinischer Pfade

Zentrale Anforderung	**Resultierende Teilanforderung**		**iMedOne®.DocPath**
1. Entwicklung und Ausführung klinischer Pfade (siehe Kapitel 2.1.5)	1.1	Klinische Pfade müssen für Fachanwender modellierbar sein	**Ja**
	1.2	Optimierung des Grades an Wiederverwendbarkeit durch formale Methoden zur Verstetigung des Prozesswissens	**Nein**
	1.3	Semantische Abhängigkeiten zwischen Prozessteilen müssen formalisiert bei der Modellierung berücksichtigt werden	**Nein**
	1.4	Die Modellierung unterschiedlicher Prozessteile kann gleichzeitig und gleichartig erfolgen	**Nein**

	1.5	Die Ausführung klinischer Pfade nach der Modellierung muss mit unterschiedlichen WfMS möglich sein	**Nein**
	1.6	Es besteht die Möglichkeit, klinische Pfade in unterschiedliche Darstellungsformen zu überführen	**Nein**

Obwohl die Lösung für die Krankenhäuser der Knappschaft gut funktioniert, ist sie nicht ohne weiteres auf andere Einrichtungen übertragbar. Grund dafür ist vor allem die enge Integration mit dem KIS iMedOne®. Darüber hinaus fehlt dem Ansatz ein formales Konzept, das die korrekte Komposition und fehlerfreie Ausführbarkeit der Pfadelemente gewährleistet. Der Mangel an einschränkenden Formalismen erleichtert jedoch andererseits die flexible Gestaltung klinischer Pfade. So können Fachexperten je nach Belieben Pfadelemente einfügen, löschen oder verschieben.

3.1.2 Klinische Pfade am ev. Krankenhaus Königin Herzberge

Noch früher als die KBS, nämlich bereits 2001 begann das ev. Krankenhaus Königin Herzberge in Berlin mit der IT-gestützten Anwendung klinischer Pfade [Tenckhoff 2003]. Da auch Interesse von anderen Gesundheitseinrichtungen bestand, wurde die Lösung als Internetplattform ClinPath entwickelt. Auf Basis dieser Plattform können Einrichtungen selbständig Pfade intern oder sektorübergreifend aufbauen. Derzeit sind über 600 Pfade bzw. Pfadmodule von über 35 Einrichtungen auf der Plattform hinterlegt; der Grad der Veröffentlichung wird von jeder Einrichtung selbst bestimmt. Die Pfade basieren auf einzelnen Modulen, wie z. B. dem Modul Gastrokopie, die flexibel als Bausteine in unterschiedlichen Pfaden verwendet werden können. Die korrekte Komposition der Module durch formale Deklaration und Auswertung semantischer Beziehungen wird nicht unterstützt. Die Pfade sind nicht fix an ein KIS bzw. WfMS gebunden, sondern einzelne Elemente lassen sich gegebenenfalls im Intranet der jeweiligen Einrichtung ausführen; dies umfasst z. B. die Bestätigung der Abarbeitung einzelner Schritte durch die Verantwortlichen und die Unterstützung von Anwendungsfunktionen zur Erstellung von Untersuchungsanforderungen oder E-Mailversand. ClinPath gewährleistet die Ausführbarkeit klinischer Pfade jedoch nicht generell, sondern stellt diesen Aspekt in die Verantwortung der IT-Spezialisten der jeweiligen Institution; das bedeutet auch, dass die Pfade nicht direkt nach ihrer Erstellung ausführbar sind, sondern dafür nachträglich Maßnahmen seitens des IT-Personals erforderlich sind. Die nachfolgende Tabelle zeigt die Untersuchungsergebnisse im Überblick.

Tabelle 6 Ergebnisse der Analyse von ClinPath zur Entwicklung und Ausführung klinischer Pfade

Zentrale Anforderung		**Resultierende Teilanforderung**		**ClinPath**
1.	Entwicklung und Ausführung klinischer Pfade (siehe Kapitel 2.1.5)	1.1	Klinische Pfade müssen für Fachanwender modellierbar sein	**Ja**
		1.2	Optimierung des Grades an Wiederverwendbarkeit durch formale Methoden zur Verstetigung des Prozesswissens	**Nein**
		1.3	Semantische Abhängigkeiten zwischen Prozessteilen müssen formalisiert bei der Modellierung berücksichtigt werden	**Nein**

1.4	Die Modellierung unterschiedlicher Prozessteile kann gleichzeitig und gleichartig erfolgen	**Nein**
1.5	Die Ausführung klinischer Pfade nach der Modellierung muss mit unterschiedlichen WfMS möglich sein	**Nein**
1.6	Es besteht die Möglichkeit, klinische Pfade in unterschiedliche Darstellungsformen zu überführen	**Nein**

Insgesamt können die ersten positiven Praxisbeispiele nicht über die Tatsache hinwegtäuschen, dass die Entwicklung wirkungsvoller klinischer Pfade noch am Anfang steht. So sind auch die Potentiale klinischer Pfade im Hinblick auf Effizienz- und Effektivitätssteigerungen noch nicht ansatzweise erschlossen [Salfeld et al. 2009][28]. Weil vor allem mit der Entwicklung papierloser, elektronischer Pfade erst begonnen wurde, entsprechen die Anpassungen an KIS, wie z. B. im Fall der KBS, noch stark zweckgebundenen, auftragsgetriebenen und proprietären Entwicklungen. Sämtliche Konzepte sind meist tief mit dem zugrunde liegenden KIS verwoben und nur schwer auf andere Einrichtungen und andere Systeme übertragbar, falls dies überhaupt gewünscht ist. Auch das macht die flächendeckende Einführung klinischer Pfade so schwierig. Alternative Ansätze, wie ClinPath, erlauben zwar die KIS-unabhängige Entwicklung klinischer Pfade; die Prozessdefinitionen lassen sich aber nur dann in einer Einrichtung ausführen, wenn einzelne Pfadelemente von IT-Spezialisten nachträglich mit internen Anwendungskomponenten verknüpft werden. Darüber hinaus finden die praktischen Entwicklungen häufig vollkommen unabhängig von den wissenschaftlichen Erkenntnisprozessen statt, deren formale Konzepte sich zwar äußerst nutzbringend einsetzen ließen, die aufgrund ihrer Generezität und Praxisferne für den klinischen Sektor jedoch oftmals verworfen werden.

3.1.3 Theorie der kompositionalen Systemmodellierung

Beide praktischen Systemlösungen, iMedOne®.DocPath und ClinPath, zerlegen klinische Pfade in einzelne Module, die anschließend wieder zu vollständigen Prozessen zusammengesetzt, komponiert werden können. Dieser Ansatz geht zurück auf das Prinzip der kompositionalen Modellierung, der 1991 bereits von Falkenhainer und Forbus für den Anwendungsbereich der Mechanik entwickelt wurde [Falkenhainer & Forbus 1991]. Den Autoren zufolge ist das traditionelle Vorgehen, nach dem für jedes Szenario ein spezifisches Modell entworfen werden muss, zu aufwändig und fehleranfällig; z. B. kann auf diese Weise nicht automatisch geprüft werden, ob ein Modell grundsätzlichen Annahmen und Regeln einer Domäne gerecht wird. Aus diesem Grund wird der Entwurf eines übergeordneten Domänenmodells motiviert, das grundlegende Vorgaben für eine Menge von verwandten Systemen aufstellt. Jede Realisierung eines speziellen Szenarios basiert dann auf dem gemeinsamen Domänenmodell, das gegebenenfalls ergänzt werden muss, um den Anforderungen des neuen Szenarios zu entsprechen.

Bei Systemlösungen wie iMedOne®.DocPath und ClinPath besteht das Domänenmodell bisher im Wesentlichen aus der Information, welche Elemente zur Komposition zur Verfügung

[28] Siehe Salfeld et al. 2009, S. 61

stehen. Durch die Aggregation von Elementen zu Modulen, die selbst wieder als Gesamtmodul in klinische Pfade eingebunden werden können, oder durch Spezifikation von Vorgängerrelationen werden Regeln zur Komposition bisher nur rudimentär umgesetzt. Ohne einen zugrunde liegenden Formalismus sind der konsequente Aufbau eines solchen Domänenmodells und die Ableitung konformer Szenarienmodelle auch nicht möglich.

Darüber hinaus entwickeln Falkenhainer und Forbus einen Ansatz, wie auf Basis einer formalen Szenariobeschreibung, eines übergeordneten Domänenmodells und einer Anfrage zum gewünschten Verhalten, Vorschläge für kohärente und am besten geeignete Szenarienmodelle automatisch generiert werden können. Das Konzept wurde an einem Lernsystem über thermodynamische Zusammenhänge erprobt, um ausgehend von dem im Domänenmodell hinterlegten Wissen Antworten zu generieren, die die Lernenden aktuell interessieren. Obwohl sich dieser Ansatz nicht 1:1 auf klinische Pfade übertragen lässt, liefert er ein theoretisches Fundament für das von den praktischen Lösungsansätzen ebenfalls verfolgte Prinzip der Modularisierung und prozessorientierten Komposition.

3.2 Ansätze zur Unterstützung manueller Abweichungsmaßnahmen

In diesem Kapitel sollen wissenschaftliche Lösungskonzepte vorgestellt und untersucht werden, die formale Methoden zur Prozessmodellierung definieren und darüber hinaus flexible Abweichungen unterstützen. Der Bedarf zur Ad-hoc-Anpassung auch nach der Instanziierung von Prozessdefinitionen wird in der Literatur vor allem unter dem Aspekt des Auftretens eines Ausnahmefalls erörtert [Strong & Miller 1995, Reichert & Dadam 1998, Russell et al. 2006]. Das Auftreten einer Varianz von klinischen Pfaden, die das Treffen manueller Abweichungsmaßnahmen erforderlich macht, ist jedoch nicht zwingend mit der Situation eines Ausnahmefalles gleichzusetzen; denn dann würde jede Nebendiagnose oder individuelle Entscheidung des Patienten, die nicht im Standardprozess berücksichtigt ist, als Ausnahme deklariert werden. Eine Ausnahme bezeichnet ein Ereignis, das den ursprünglichen Erwartungen widerspricht und mit dem selten zu rechnen ist. Dies korrespondiert jedoch nicht mit der Definition von Varianz in Zusammenhang mit klinischen Pfaden (siehe Kapitel 2.2.2).

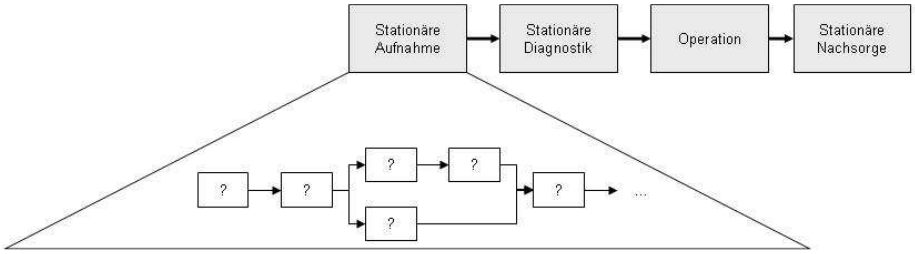
Demzufolge kann sich bei der Patientenversorgung also im Vergleich zu anderen Domänen die Notwendigkeit einer Ad-hoc-Anpassung des Prozesses vor oder nach seiner Instanziierung auch aufgrund von nicht ausnahmebedingten Ereignissen und damit häufiger ergeben. Somit stellt sich an Lösungsansätze zur Realisierung klinischer Pfade umso mehr die Anforderung, dass Abweichungsmaßnahmen möglichst einfach, effizient und ohne großen Aufwand für das Krankenhauspersonal bewältigt werden können. Ansonsten steigt die Wahrscheinlichkeit, dass Anwender das System umgehen, statt die notwendigen Änderungen in den Prozess einzupflegen [Strong & Miller 1995].

Vorab kann für alle Ansätze festgestellt werden, dass sie die Modellierung ausführbarer klinischer Pfade durch Fachexperten nicht unterstützen; für die Darstellung und Bedienung durch das medizinische Krankenhauspersonal sind sie ungeeignet. Dennoch bieten sie formale Methoden, die dazu beitragen, Prozessdefinitionen zu validieren und die korrekte Ausführung der davon abgeleiteten Instanzen zu gewährleisten.

3.2.1 Abstraktion von Prozessdetails

Die einfachste Möglichkeit um ein hohes Maß an Flexibilität zu erreichen und auch ad hoc Änderungen an laufenden Prozessinstanzen zuzulassen, ist die Abstraktion von Prozessdetails. Der klinische Pfad aus Abbildung 9 ist auf einem sehr hohen Abstraktionsniveau dargestellt und besteht in dieser Form lediglich aus vier aufeinander folgenden Aktivitäten. Jede dieser Aktivitäten umfasst eine Menge von Subaktivitäten, die in einer logischen Reihenfolge nacheinander oder auch parallel zueinander bearbeitet werden. Werden diese Subaktivitäten in der Prozessdefinition nicht aufgeführt, so können Patienten, die unter einer Vorerkrankung leiden oder bei denen eine Komplikation auftritt, auf Basis desselben klinischen Pfades behandelt werden; dem Krankenhauspersonal steht es frei, zusätzliche oder andere Aktivitäten auszuführen, da das WfMS keine Notiz davon nimmt.

Abbildung 9 Flexibilität durch Abstraktion von Prozessdetails



In Kapitel 2.1.4.2 wurde jedoch erläutert, dass zu den wichtigsten Aufgaben eines WfMS die Schaffung von Prozesstransparenz, die Steuerung und Koordination des Prozessablaufs, die Erleichterung der Ressourcenplanung, der Kostenkontrolle und der Durchsetzung von Qualitätsstandards zählen. Darüber hinaus sind eine kontinuierliche Pflege und Weiterentwicklung klinischer Pfade auf Basis der Varianzdokumentation nicht zu erreichen, wenn eine Unterscheidung zwischen Prozessstandard und Varianz aufgrund des hohen Abstraktionsniveaus nicht möglich ist. Sollen die Vorteile von WfMS genutzt werden, ist die simple Abstraktion von Prozessdetails, um sichtbare Ad-hoc-Modifikationen zu vermeiden, also keine tragfähige Lösung.

Mit zunehmendem Detaillierungsgrad bei der Modellierung klinischer Pfade gewinnt das WfMS an Bedeutung für die Steuerung von Behandlungsprozessen; parallel dazu steigen jedoch auch die Komplexität und die Wahrscheinlichkeit von Varianz. Je wichtiger also die Rolle des WfMS zur Koordination der Patientenversorgung wird, desto dringender werden Strategien zur Flexibilisierung und Beherrschbarkeit solcher Systeme.

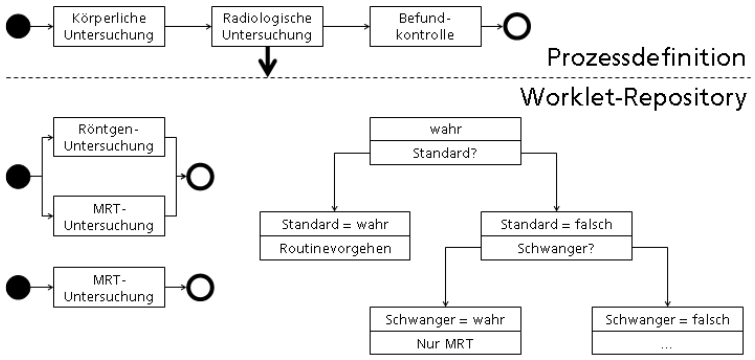
3.2.2 Late Binding von Subprozessen

Das Konzept des Late Binding zielt auf die Flexibilisierung der Prozessausführung ab, indem die konkrete Realisierung einer Aktivität erst zur Laufzeit determiniert wird. Der Ansatz wird im Bereich der Programmierung häufig verwendet und in Zusammenhang mit WfMS z. B. bei der

dynamischen Allokation von Ressourcen genutzt [Weske 2007]. Mit Worklets wird ein Lösungskonzept für die dynamische Operationalisierung von Prozessaktivitäten vorgestellt [Adams et al. 2005, Adams et al. 2006, ter Hofstede et al. 2010][29]. Bei Worklets handelt es sich um eigenständige Subprozesse, die in Abhängigkeit des situationsspezifischen Kontexts, in dem die Prozessinstanz ausgeführt wird, ad hoc selektiert werden können. Im Gegensatz zur Prozessflexibilisierung durch Abstraktion, kann das WfMS die Ausführung des Prozesses weiterhin steuern, die Realisierung der Aktivitäten wird jedoch dynamisch festgelegt.

Anstelle von vollständig definierten und operationalisierten Aktivitäten enthält die Prozessdefinition lediglich Vorlagen, die auf eine Menge von Worklets und ein Regelwerk in Form von RDR-Regeln (Ripple Down Rules) verweisen. Die RDR Methode ist formal und hat sich in einer Reihe von Anwendungsfällen bereits etablieren können [Scheffer 1996]. Die Regeln werden in einer binären Baumstruktur angeordnet und ausgehend von der Wurzel durchlaufen, bis ein Blattknoten erreicht wird. Der Weg durch den Binärbaum richtet sich nach der Evaluation der Regeln nach »wahr« oder »falsch«. Ergibt die Auswertung der Regel im Blattknoten den Wert »falsch«, so wird der Elternknoten als Ergebnis zurückgeliefert. Adams et al. verbinden die Ergebnisse der RDR-Methode nun mit der Selektion entsprechender Worklets. Sieht der Prozessablauf z. B. normalerweise eine Röntgenuntersuchung und ein MRT vor, so kann im Fall einer Schwangerschaft das Worklet gewählt werden, das nur eine MRT-Untersuchung umfasst. Die folgende Abbildung illustriert das Konzept.

Abbildung 10 Selektion eines Worklets zur Realisierung einer Aktivität durch RDR-Regeln



Generell lässt sich die Worklet-Menge zur Laufzeit beliebig erweitern. Stellt der Anwender z. B. fest, dass weder das Worklet für den Standardfall noch das Worklet für schwangere Patientinnen für die Behandlung seines Patienten geeignet ist, kann er die Regelmenge dynamisch erweitern und ein zusätzliches Worklet erstellen. Dieses Worklet wird zusammen mit allen anderen Worklets in einem Repository verwaltet und steht dort zukünftig ebenfalls zur Verfügung. Die Modellierung von Worklets setzt je nach Komplexität des notwendigen Subprozesses voraus, dass das Krankenhauspersonal über das technische Verständnis und die Übung verfügt, Worklets zu erstellen. Angesichts des Zeitdrucks in Kliniken und der Tatsache, dass es sich bei der Prozessmodellierung um eine fachfremde Aufgabe handelt, muss die Möglichkeit der flexiblen Erweiterung der Worklet-Menge durch die Anwender jedoch bezwei-

[29] Siehe ter Hofstede et al. 2010 S. 124ff

felt werden. Die nachfolgende Tabelle zeigt die Ergebnisse der Untersuchung des Late-Binding (speziell: des Worklet Ansatzes) auf seine Eignung zur Unterstützung manueller Abweichungsmaßnahmen.

Tabelle 7 Ergebnisse der Analyse von Worklets zur Unterstützung manueller Abweichungsmaßnahmen

Zentrale Anforderung		**Resultierende Teilanforderung**		**Worklets**
2.	Unterstützung manueller Abweichungsmaßnahmen	2.1	Manuelle Abweichungsmaßnahmen vom klinischen Pfad müssen für Fachanwender durchführbar sein (siehe Kapitel 2.2.3.1, 2.2.3.4.1, 2.2.3.4.4)	**Ja** (sofern vorhandene Worklets genutzt werden können)
		2.2	Fachanwender und WfMS müssen klar zwischen Standardprozess und vom Standard abweichenden Maßnahmen unterscheiden können (siehe Kapitel 2.2.3.1, 2.2.3.2)	**Ja** (abhängig von der Gestaltung der RDR-Baums)
		2.3	Die Wahl der Abweichungsmaßnahmen muss in Abhängigkeit des Behandlungsfalls variieren können (siehe Kapitel 2.2.3.3)	**Ja**
		2.4	Fachexperten müssen sowohl die Möglichkeit haben sofort als auch verzögert auf eine Varianz zu reagieren (siehe Kapitel 2.2.3.4.2)	**Nein**
		2.5	Abweichungsmaßnahmen können zu unterschiedlichen Zeitpunkten und an unterschiedlichen Stellen im Prozess wiederholt werden (siehe Kapitel 2.2.3.4.3)	**Ja**
		2.6	Dieselben Abweichungsmaßnahmen lassen sich gleichzeitig an mehreren Stellen des Prozesses durchführen (siehe Kapitel 2.2.3.4.5)	**Nein**
		2.7	Semantische Abhängigkeiten zwischen Abweichungsmaßnahmen müssen bei ihrer Auswahl berücksichtigt werden (siehe Kapitel 2.2.3.4.6)	**Nein**
		2.8	Dieselben Abweichungsmaßnahmen können unabhängig von bestimmten klinischen Pfaden durchgeführt werden (siehe Kapitel 2.2.3.4.7)	**Ja**
		2.9	Die Ausführung des klinischen Pfades muss auch nach der Durchführung manueller Abweichungsmaßnahmen möglich sein (siehe Kapitel 2.1.5)	**Nein**

Durch die intuitive Gestaltung des RDR-Baums werden Fachanwender bei der Auswahl der Worklets unterstützt und können Abweichungsmaßnahmen selbständig durchführen. Dies gilt zumindest solange existierende Worklets zur Auswahl bereitstehen. Die Unterscheidbarkeit zwischen Standardvorgehen und Abweichung wird im Worklet-Konzept nicht direkt adressiert. Allerdings lässt sich der RDR-Baum so gestalten, dass zwischen Standard und Varianz differenziert werden kann. Da das Worklet-Repository eine Menge von Worklets umfassen kann, können die Abweichungsmaßnahmen auch je nach Behandlungsfall unterschiedlich ausfallen. Auch ist es möglich dieselben Maßnahmen an unterschiedlichen Stellen durch die Einbindung des entsprechenden Worklets zu wiederholen. Gleichzeitig kann das Repository unabhängig von klinischen Pfaden genutzt werden, sofern die Worklet-Aktivitäten nicht mit den Aktivitäten des Prozesses, in den sie integriert werden, interagieren.

Dadurch, dass die Auswahl der Worklets nur an definierten Stellen im Prozess möglich ist, sind der Flexibilität bei ihrer Einbindung enge Grenzen gesetzt. Ebenso verhält es sich mit der Wiederholung von Abweichungsmaßnahmen an unterschiedlichen Stellen und zu unterschiedlichen Zeiten. Um dies zu unterstützen, müssten Prozessdefinitionen nahezu überall die Auswahl und Einbettung von Worklets erlauben. Auch die gleichzeitige Durchführung von Abweichungsmaßnahmen an verschiedenen Positionen im Prozess ist nicht möglich; ein Worklet kann erst dann gewählt werden, wenn die Ausführung die entsprechende Stelle im Prozess erreicht hat. Semantische Abhängigkeiten zwischen Abweichungsmaßnahmen lassen sich gar nicht definieren. Sofern Fachexperten selbst Worklets definieren können ist die Ausführbarkeit des Prozesses nach einer Anpassung nicht gesichert.

3.2.3 Late Modeling von Prozessteilen

Während es mit »Late Binding« ausschließlich möglich ist, die Realisierung von Subprozessen zur Laufzeit festzulegen, ermöglichen »Late Modeling« Ansätze die Ausführung von Prozessen, die nur teilweise modelliert sind und zur Laufzeit dynamisch vervollständigt werden können. Die entsprechenden Lösungskonzepte unterscheiden sich nicht nur im Hinblick auf die Workflow-Aspekte, die unspezifiziert bleiben, sondern sie räumen den Anwendern auch verschiedene Freiheitsgrade bei der Vervollständigung der Prozessinstanzen ein [Hagemeyer et al. 1997].

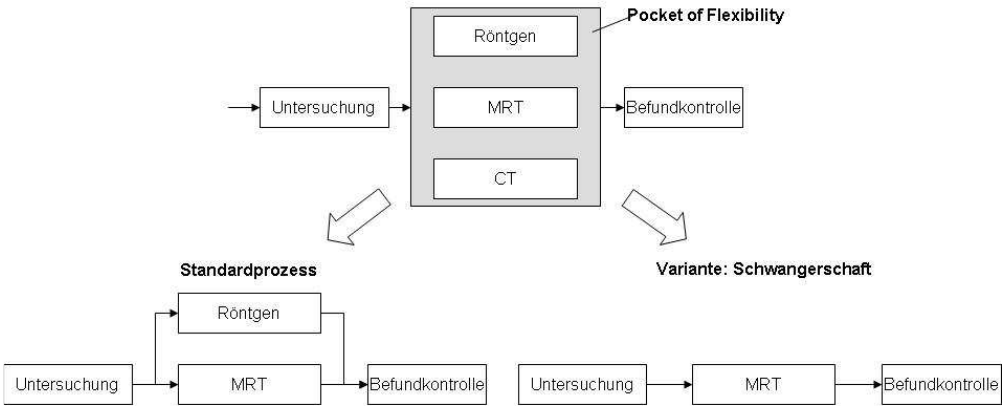
Bei FUNSOFT-Netzen handelt es sich um eine Prozessmodellierungssprache, die Petri Netze u. a. um die Möglichkeit zur Integration von unspezifizierten Regionen in Prozessdefinitionen erweitert [Gruhn 1991]. FUNSOFT-Netze werden unter Nutzung der WfM Umgebung COREMAN [Deiters et al. 1995] (früher: MELMAC [Deiters et al. 1994]) erstellt und ausgeführt. Änderungen an Teilprozessen während der Laufzeit werden durch entsprechende Ereignisse ausgelöst; z. B. wenn der Bestand an Medikamenten nicht ausreicht, um einen Patienten zu versorgen. Eine adäquate Änderung der Prozessinstanz würde das Einfügen einer Aktivität oder eines Subprozesses zur Bestellung des entsprechenden Medikaments bewirken. Generell können Modifikationen das Einfügen, Ändern oder Löschen von Prozessteilen zur Folge haben. Während der Adaption können die Aktivitäten, die von dem zu ändernden Bereich unabhängig sind, weiter ausgeführt werden. Um variable Bestandteile innerhalb einer Prozessdefinition von nicht änderbaren Bereichen zu unterscheiden, führt Deiters den so genannten »Change View« ein, der nur die variablen Teile umfasst [Deiters 1992].

Bei MOBILE handelt es sich um ein modulares WfMS, dessen Module mit den verschiedenen Workflow Aspekten, wie z. B. verhaltens- oder informationsbezogener Aspekt, korrespondieren [Jablonski et al. 1997a]. Prozessdefinitionen für MOBILE werden in einer eigenen Workflow Modellierungssprache MOMO (Mobile Modeling Language) erstellt, die das offen lassen bestimmter Bereiche zur Modellierungs- und Instanziierungszeit erlaubt, wenn notwendige Aktivitäten oder Subprozesse erst während der Ausführung festgelegt werden können [Horn & Jablonski 1998, Heinl et al. 1999]. Zusätzlich können zu jeder Prozessdefinition Zielbeschreibungen hinterlegt werden; diese Ziele dienen als Constraints, die die Möglichkeiten des nachträglichen Vervollständigens unspezifizierter Prozessbereiche einschränken. So darf die Prozessinstanz z. B. nicht durch eine Fülle von Aktivitäten erweitert werden, die dazu führen, dass die maximale Verweildauer eines Patienten im Krankenhaus überschritten wird. Der Ansatz definiert allerdings keine Kriterien, die der Gewährleistung der strukturellen Korrektheit von Prozessen dienen. Andere Lösungskonzepte lassen auch den organisatorischen oder operati-

onalen Aspekt (Late Binding) bis zur Laufzeit unspezifiziert [Han et al. 1996, Liu & Pu 1997, Chiu et al. 1999].

Sadiq et al. charakterisieren klinische Prozesse als typisches Anwendungsbeispiel für semi-strukturierte Prozesse mit unspezifizierten Regionen [Sadiq et al. 2001]; sie argumentieren, dass klinische Prozesse einerseits aus stark strukturierbaren Bestandteilen, wie z. B. administrativen Abläufen bestehen, und zum anderen individuell geprägte diagnostische und therapeutische Vorgehenswiesen enthalten. Auf klinische Pfade trifft diese Beschreibung nicht ganz zu, da sie auch auf die Standardisierung medizinischer Prozesse abzielen, mit der Option im Bedarfsfall eine besser geeignete Handlungsweise zu wählen. Sadiq et al. führen das Konzept der Pockets of Flexibility (PoF) ein. PoF können wie normale Aktivitäten in eine Prozessdefinition eingefügt werden; sie umfassen einen Satz von Prozessfragmenten, die wiederum aus einer einzelnen Aktivität oder einem Subprozess bestehen. Sobald eine Prozessdefinition instanziiert wurde und die Prozessausführung ein PoF erreicht, werden die Prozessfragmente je nach individueller Situation ausgewählt und miteinander kombiniert. Es besteht auch die Möglichkeit, zur Laufzeit Prozessfragmente selbst zu modellieren und dem PoF hinzuzufügen. Die nachfolgende Abbildung demonstriert die Einbettung eines PoFs in die Prozessdefinition und die Ableitung unterschiedlicher Prozessinstanzen.

Abbildung 11 Ad hoc Vervollständigung unspezifizierter Regionen der Prozessdefinition zur Laufzeit



Möglichkeiten zur Kombination von Prozessfragmenten können wiederum durch Constraints eingeschränkt sein [Sadiq et al. 2001, Mangan & Sadiq 2002, Mangan & Sadiq 2003, Sadiq et al. 2005]. Wird die Menge der Prozessfragmente im Voraus eingeschränkt, spricht man auch von »Late Composition«. Durch die Festlegung der Art und Weise, wie der Prozess vervollständigt werden kann, kann die Ausführbarkeit des Prozesses sichergestellt werden.

Der Late Modeling Ansatz räumt Anwendern prinzipiell dieselben Freiheiten bei der Durchführung von Abweichungsmaßnahmen wie bei der initialen Prozessmodellierung ein. Einschränkungen existieren nur im Hinblick auf die Regionen im Prozess, die nachträglich mit Inhalten gefüllt werden können. Durch die Spezifikation von Constraints ist es jedoch auch möglich, die Fülle an Optionen beim Late Modeling bzw. Late Composition für die Anwender einzuschränken. Prinzipiell lassen sich dieselben Änderungsmaßnahmen auch an unterschiedlichen Pfaden durchführen, sofern dort unspezifizierte Bereiche angegeben sind. Ähnlich wie bei Worklets muss es also für jeden Pfad im Voraus eine genaue Festlegung der Positionen geben, an denen Abweichungen vom Standard möglich sind.

Da die Ansätze nicht darauf abzielen, Fachanwender ohne technische Kenntnisse in die Lage zu versetzen, Prozesse entwickeln zu können, gilt dies auch im Fall der Durchführung von Änderungsmaßnahmen. Bei Late Modeling wird generell von der Annahme ausgegangen, dass es für die unspezifizierten Bereiche eben kein Standardvorgehen gibt; daher besteht auch nicht die Möglichkeit, zwischen Standard und Varianz zu unterscheiden. Durch Festlegung der unspezifizierten Bereiche im Voraus, lassen sich Entscheidungen im Hinblick auf ihre Gestaltung immer nur dann treffen, wenn die Prozessausführung den entsprechenden Bereich erreicht hat. Die Flexibilität hinsichtlich des Zeitpunkts der Abweichung ist also stark eingeschränkt. Das gleichzeitige Durchführen derselben Maßnahmen an unterschiedlichen Stellen im Prozess ist ebenfalls nicht vorgesehen. Da im Rahmen von Late Modeling prinzipiell völlig neue Prozessaktivitäten erstellt werden können, ist die Ausführbarkeit der Prozessdefinition nach einer Anpassung nicht gesichert; dies ist nur möglich, sobald IT-Spezialisten den einzelnen Schritten Anwendungskomponenten zugeordnet und die Konsistenz der Datenflüsse hergestellt haben. Vorab kann das nur im Rahmen des Late Composition Ansatzes erfolgen.

Tabelle 8 Ergebnisse der Analyse von Ansätzen des Late Modeling zur Unterstützung manueller Abweichungsmaßnahmen

Zentrale Anforderung		**Resultierende Teilanforderung**		**COR-MAN/MOBILE/POF**
2.	Unterstützung manueller Abweichungsmaßnahmen	2.1	Manuelle Abweichungsmaßnahmen vom klinischen Pfad müssen für Fachanwender durchführbar sein (siehe Kapitel 2.2.3.1, 2.2.3.4.1, 2.2.3.4.4)	**Nein**
		2.2	Fachanwender und WfMS müssen klar zwischen Standardprozess und vom Standard abweichenden Maßnahmen unterscheiden können (siehe Kapitel 2.2.3.1, 2.2.3.2)	**Nein**
		2.3	Die Wahl der Abweichungsmaßnahmen muss in Abhängigkeit des Behandlungsfalls variieren können (siehe Kapitel 2.2.3.3)	**Ja**
		2.4	Fachexperten müssen sowohl die Möglichkeit haben sofort als auch verzögert auf eine Varianz zu reagieren (siehe Kapitel 2.2.3.4.2)	**Nein**
		2.5	Abweichungsmaßnahmen können zu unterschiedlichen Zeitpunkten und an unterschiedlichen Stellen im Prozess wiederholt werden (siehe Kapitel 2.2.3.4.3)	**Ja**
		2.6	Dieselben Abweichungsmaßnahmen lassen sich gleichzeitig an mehreren Stellen des Prozesses durchführen (siehe Kapitel 2.2.3.4.5)	**Nein**
		2.7	Semantische Abhängigkeiten zwischen Abweichungsmaßnahmen müssen bei ihrer Auswahl berücksichtigt werden (siehe Kapitel 2.2.3.4.6)	**Ja** (mit Constraints)
		2.8	Dieselben Abweichungsmaßnahmen können unabhängig von bestimmten klinischen Pfaden durchgeführt werden (siehe Kapitel 2.2.3.4.7)	**Ja**
		2.9	Die Ausführung des klinischen Pfades muss auch nach der Durchführung manueller Abweichungsmaßnahmen möglich sein (siehe Kapitel 2.1.5)	**Nein** (nur mit Late Composition)

3.2.4 Deklarative Prozessmodellierung

Während mit dem Late Modeling Ansatz imperative bzw. prozedurale Prozessmodellierungssprachen lediglich um deklarative Anteile ergänzt werden, existieren auch WfMS, bei denen die Prozessmodellierung komplett auf dem deklarativen Paradigma beruht. Ein Beispiel für ein WfMS, das die Instanziierung und Ausführung deklarativer Prozessdefinitionen ermöglicht, ist Declare [Pesic et al. 2007]. Mit Declare soll zum einen die Notwendigkeit struktureller Ad-hoc-Prozessänderungen zur Laufzeit reduziert und zum anderen Überspezifikation vermieden werden. Überspezifikation entsteht vor allem dadurch, dass imperative Prozessmodellierungssprachen die Vorgabe einer Ausführungsreihenfolge von Aktivitäten durch die Definition von Kontrollflüssen erzwingen, auch wenn dies im konkreten Anwendungsfall nicht sinnvoll ist. Aus diesem Grund gibt es bei Declare keine Kontrollflüsse. So wird lediglich festgelegt, was im Rahmen eines Prozesses getan werden soll, aber nicht wie es zu erfolgen hat. Prozessdefinitionen liegen bei Declare als Constraint Modelle vor. Diese bestehen aus einer Menge von Aktivitäten, einer Menge von Constraints und einer Funktion, die einen Constraint als verpflichtend oder optional deklariert. Ein Constraint kann z. B. vorschreiben, dass eine Aktivität mindestens einmal ausgeführt werden muss oder dass eine Aktivität nur nach einer anderen Aktivität vorkommen darf. Während verpflichtende Constraints unbedingt einzuhalten sind, dürfen optionale Constraints verletzt werden. Die formale Basis für Prozessmodellierungssprachen für Declare (z. B. ConDec [Pesic & van der Aalst 2006] und DecSerFlow [van der Aalst & Pesic 2006]) bildet die Lineare Temporale Logik (LTL) [Clarke Jr. et al. 1999]. Graphische Prozessdefinitionen, die einer Menge von Constraints entsprechen, werden geeignet in LTL-Terme transformiert. Für die LTL-Terme kann ein endlicher Automat angegeben werden, dessen Ausführung simuliert und dessen Korrektheit mit formalen Methoden verifiziert werden kann. Im Rahmen von Declare wird zu diesem Zweck das Spin-Tool verwendet [Holzmann 2003, Clarke Jr. et al. 1999]. Um die Analysierbarkeit von Prozessdefinitionen, die mit deklarativen Modellierungssprachen erstellt wurden noch zu erhöhen, stellt Fahland einen Ansatz vor, wie deklarative Prozessdefinitionen in Petri Netze überführt werden können [Fahland 2007]. Mulyar et al. untersuchen die Eignung der deklarativen Prozessmodellierung zur Abbildung medizinischer Leitlinien [Mulyar et al. 2008].

Polyvyanyy und Weske präsentieren einen Modellierungsansatz für flexible Prozessgraphen (FPG), der auf Hypergraphen basiert [Polyvyanyy & Weske 2009]. Ein Hypergraph ist ein generalisierter Graph, dessen Kanten beliebig große Mengen von Knoten miteinander verbinden können [Berge 1985, Berge 1989]. Ähnlich wie bei Declare wird also auf eine vollständige Spezifikation durch die exakte Vorgabe eines Kontrollflusses verzichtet. Das hypergraph-basierte Prozessmodell lässt aufgrund seiner Strukturierung mehrere Möglichkeiten der Ausführung zu. Um die Auswahl von Aktivitäten, die über eine Hyperkante miteinander verknüpft sind, einzuschränken, können die Kanten mit logischen Operatoren (AND, OR, XOR) versehen werden. Neben der formalen Definition von FPG geben die Autoren auch eine formale Ausführungssemantik an. Nach der Instanziierung des FPG, können die Anwender ähnlich wie bei Declare flexibel zwischen den aktuell frei geschalteten Aktivitäten auswählen.

Für die Anwender gibt es im Hinblick auf die Art und Weise, in der deklarative klinische Pfade abgearbeitet werden, keinen Unterschied zwischen Standardvorgehen und Abweichung. Typus, Zeitpunkt, Position und Häufigkeit der Abweichungsmaßnahmen lassen sich sehr flexibel festlegen. Durch logische Operatoren bei FPG und Constraint-Relationen in Declare lassen sich auch semantische Beziehungen zwischen einzelnen oder Gruppen von Maßnahmen definieren. Declare und FPG zielen vor allem darauf ab, die Ausführung von Prozessen zu flexibilisieren; dabei kommt im Hinblick auf klinische Pfade jedoch der Standardisierungsaspekt zu

kurz. Die Tatsache, dass eine Unterscheidung zwischen Standardprozess und Abweichung nicht möglich ist, erleichtert zwar die Bedienbarkeit des Systems, widerspricht jedoch dem Ziel klinischer Pfade, Behandlungsprozesse durch die klare Vorgabe eines Standards stärker zu vereinheitlichen. In Declare besteht zwar die Möglichkeit durch Spezifikation optionaler Constraints Abweichungen zu signalisieren; mit zunehmender Anzahl von Abweichungsoptionen ist die Komplexität der Prozessmodelle dann allerdings kaum mehr beherrschbar. Die gleichzeitige Durchführung derselben Änderungsmaßnahmen an unterschiedlichen Stellen im Prozess ist nicht möglich, schon allein da die Reihenfolge, in der die Aktivitäten abgearbeitet werden, nicht eindeutig definiert ist. Sämtliche Änderungsmaßnahmen sind Bestandteil desselben klinischen Pfades und lassen sich daher nicht pfadunabhängig angeben. Da Prozessmodelle in Declare und FPG bislang überhaupt keine Datenflüsse vorsehen, kann die Verfügbarkeit der von den Anwendungskomponenten benötigten Datenobjekte und damit die fehlerfreie Ausführbarkeit des Prozesses nicht von Seiten des Systems gewährleistet werden. Die Zusammenfassung der Analyseergebnisse wird in der nachfolgenden Tabelle präsentiert.

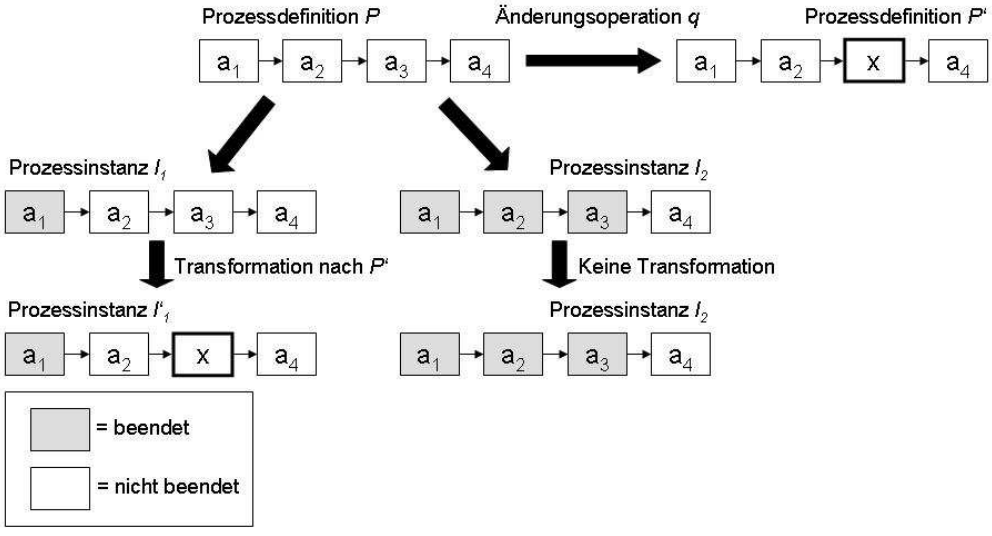
Tabelle 9 Ergebnisse der Analyse von Declare zur Unterstützung manueller Abweichungsmaßnahmen

Zentrale Anforderung		**Resultierende Teilanforderung**		**Declare/FPG**
2.	Unterstützung manueller Abweichungsmaßnahmen	2.1	Manuelle Abweichungsmaßnahmen vom klinischen Pfad müssen für Fachanwender durchführbar sein (siehe Kapitel 2.2.3.1, 2.2.3.4.1, 2.2.3.4.4)	**Ja**
		2.2	Fachanwender und WfMS müssen klar zwischen Standardprozess und vom Standard abweichenden Maßnahmen unterscheiden können (siehe Kapitel 2.2.3.1, 2.2.3.2)	**Nein**
		2.3	Die Wahl der Abweichungsmaßnahmen muss in Abhängigkeit des Behandlungsfalls variieren können (siehe Kapitel 2.2.3.3)	**Ja**
		2.4	Fachexperten müssen sowohl die Möglichkeit haben sofort als auch verzögert auf eine Varianz zu reagieren (siehe Kapitel 2.2.3.4.2)	**Ja**
		2.5	Abweichungsmaßnahmen können zu unterschiedlichen Zeitpunkten und an unterschiedlichen Stellen im Prozess wiederholt werden (siehe Kapitel 2.2.3.4.3)	**Ja**
		2.6	Dieselben Abweichungsmaßnahmen lassen sich gleichzeitig an mehreren Stellen des Prozesses durchführen (siehe Kapitel 2.2.3.4.5)	**Nein**
		2.7	Semantische Abhängigkeiten zwischen Abweichungsmaßnahmen müssen bei ihrer Auswahl berücksichtigt werden (siehe Kapitel 2.2.3.4.6)	**Ja** (vor allem Declare)
		2.8	Dieselben Abweichungsmaßnahmen können unabhängig von bestimmten klinischen Pfaden durchgeführt werden (siehe Kapitel 2.2.3.4.7)	**Nein**
		2.9	Die Ausführung des klinischen Pfades muss auch nach der Durchführung manueller Abweichungsmaßnahmen möglich sein (siehe Kapitel 2.1.5)	**Nein**

3.2.5 Nutzung von Änderungsprimitiven

Eine wesentliche Einschränkung bei allen bisher betrachteten Lösungsansätzen besteht darin, dass sie keine oder nur in bestimmten Bereichen strukturelle Prozessänderungen zulassen; d. h. Änderungen sind nur an den in der Prozessdefinition vorgesehenen Stellen möglich. Es existieren jedoch bereits WfMS, die die Durchführung von ad hoc Änderungen auch an beliebigen Positionen erlauben; dazu zählen z. B. CAKE2 (Collaborative Agent-based Knowledge Engine) [Minor et al. 2007a, Minor et al. 2007b] und WASA2 (Workflow-based Architecture to support Scientific Applications) [Weske 2000]. Bei CAKE2 handelt es sich um ein WfMS, das dynamische, lang laufende Prozesse unterstützen soll und bereits mit Szenarien im Bereich Banken, Projektmanagement und bei der Chipproduktion erprobt wurde. WASA2 basiert auf einem formalen Wf-Metamodell, das speziell unter dem Aspekt der Prozessflexibilisierung entworfen wurde. Ausgehend von Anpassungen an Prozessdefinitionen wird unter Nutzung eines mathematischen Formalismus evaluiert, ob die Änderung auch auf die Prozessinstanz übertragbar ist. Voraussetzung dafür ist, dass die bisherige Ausführungshistorie einer Prozessinstanz auf der Grundlage der modifizierten Prozessdefinition erzeugt werden kann. Die nachfolgende Abbildung zeigt eine Prozessdefinition *D*, die mittels Änderungsoperation *q* in eine Prozessdefinition *D'* transformiert wird. Angenommen es existieren zwei Prozessinstanzen I_1 und I_2, wobei bei I_1 Aktivität a_1 beendet ist, während die Ausführung bei I_2 bis Aktivität a_4 fortgeschritten ist. Da sich die Ausführungshistorie von I_1 auch auf Basis von *D'* simulieren lässt, ist es möglich I_1 nach *D'* zu migrieren. Dagegen kann die Ausführungshistorie von I_2 mit *D'* nicht nachgebildet werden; eine Migration ist daher nicht durchführbar.

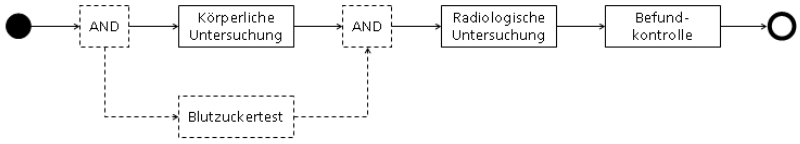
Abbildung 12 Transformation von Prozessinstanzen in Abhängigkeit ihrer Ausführungshistorie



Sowohl in CAKE2 als auch in WASA2 basieren die möglichen Prozessänderungen auf wohl definierten Änderungsoperationen, u. a. *Aktivität einfügen*, *Aktivität löschen* oder *Kontrollfluss einfügen*. Solche Änderungsprimitive machen die Durchführung von Ad-hoc-Modifikationen an Prozessinstanzen für die Anwender jedoch sehr aufwändig [Weber et al. 2009]. Die nachfolgende Abbildung illustriert vereinfacht den Ablauf der stationären Diagnostik. In dieser Phase werden Laborproben entnommen und analysiert und es wird eine radiologische Untersuchung durchgeführt. Sobald die Befunde vorliegen, ist es möglich, den Gesundheitsstatus des Patienten und sein Risikoprofil zu bewerten und die Operation zu planen. Bei Patienten, die unter

Diabetes mellitus leiden, muss neben den Standard-Laboruntersuchungen auch ein Blutzuckertest erstellt werden. Da dies im Standardablauf des klinischen Pfades nicht vorgesehen ist, wird der Test ad hoc eingefügt. In der Tabelle ist die Menge an einfachen Änderungsoperationen aufgeführt, die der Anwender vornehmen muss, um den Prozess entsprechend zu modifizieren.

Abbildung 13 Notwendige Änderungsprimitive zum parallelen Einfügen einer neuen Aktivität



Änderungsprimitive:

- addActivity(Blutzuckertest)
- addActivity(AND-Split)
- addActivity(AND-Join)
- MoveControlFlow((Start, Körperliche Untersuchung), (Start, AND-Split))
- MoveControlFlow((Körperliche Untersuchung, Radiologische Untersuchung), (Körperliche Untersuchung, And-Join))
- AddControlFlow(AND-Split, Blutzuckertest)
- AddControlFlow(Blutzuckertest, AND-Join)
- AddControlFlow(AND-Split, Körperliche Untersuchung)
- AddControlFlow(AND-Join, Radiologische Untersuchung)

Obwohl in dem Beispiel zusätzliche Datenobjekte und Änderungen am Datenfluss nicht berücksichtigt wurden, wird bereits deutlich, dass Änderungsprimitive Anwender nicht ausreichend bei manuellen Abweichungsmaßnahmen unterstützen. Außerdem führt die Anwendung der Änderungsprimitive zwangsläufig zu temporär nicht ausführbaren Prozessdefinitionen. Die Ausführbarkeit kann demnach nur in Folge automatischer Erreichbarkeitsanalysen nach Abschluss der Änderungen überprüft werden, wobei der Anwender selbst in der Pflicht ist eventuell aufgetretene Fehler zu beheben. Die automatische Validation von Prozessdefinitionen und ihren Instanzen wird von CAKE2 und WASA2 unterstützt. Tabelle 10 zeigt, welche Anforderungen an die Umsetzung von manuellen Änderungsmaßnahmen mit Hilfe von Änderungsprimitiven erfüllt werden.

Tabelle 10 Ergebnisse der Analyse von CAKE2 und WASA2 zur Unterstützung manueller Abweichungsmaßnahmen

Zentrale Anforderung		**Resultierende Teilanforderung**		**CAKE2/WASA2**
2.	Unterstützung manueller Abweichungsmaßnahmen	2.1	Manuelle Abweichungsmaßnahmen vom klinischen Pfad müssen für Fachanwender durchführbar sein (siehe Kapitel 2.2.3.1, 2.2.3.4.1, 2.2.3.4.4)	**Nein**
		2.2	Fachanwender und WfMS müssen klar zwischen Standardprozess und vom Standard abweichenden Maßnahmen unterscheiden können (siehe Kapitel 2.2.3.1, 2.2.3.2)	**Ja**
		2.3	Die Wahl der Abweichungsmaßnahmen muss in Abhängigkeit des Behandlungsfalls variieren können (siehe Kapitel 2.2.3.3)	**Ja**
		2.4	Fachexperten müssen sowohl die Möglichkeit haben sofort als auch verzögert auf eine Varianz zu reagieren (siehe Kapitel 2.2.3.4.2)	**Ja**

2.5	Abweichungsmaßnahmen können zu unterschiedlichen Zeitpunkten und an unterschiedlichen Stellen im Prozess wiederholt werden (siehe Kapitel 2.2.3.4.3)	**Ja**
2.6	Dieselben Abweichungsmaßnahmen lassen sich gleichzeitig an mehreren Stellen des Prozesses durchführen (siehe Kapitel 2.2.3.4.5)	**Nein**
2.7	Semantische Abhängigkeiten zwischen Abweichungsmaßnahmen müssen bei ihrer Auswahl berücksichtigt werden (siehe Kapitel 2.2.3.4.6)	**Nein**
2.8	Dieselben Abweichungsmaßnahmen können unabhängig von bestimmten klinischen Pfaden durchgeführt werden (siehe Kapitel 2.2.3.4.7)	**Ja**
2.9	Die Ausführung des klinischen Pfades muss auch nach der Durchführung manueller Abweichungsmaßnahmen möglich sein (siehe Kapitel 2.1.5)	**Nein**

Durch die Entwicklung des klinischen Pfades als prozedurale Prozessdefinition, liegt den Anwendern ein Standardprozess vor. Abweichungen von diesem Prozess entstehen durch die strukturelle Anpassung dieses Standards; somit ist eine klare Unterscheidung möglich. Prinzipiell können mit Hilfe der Änderungsprimitive beliebige Modifikationen an der Prozessdefinition und letztlich auch den Instanzen durchgeführt werden, sofern die vom System vorgegebenen Korrektheitsanforderungen weiterhin erfüllt sind. Auch können dieselben Abweichungsmaßnahmen für unterschiedliche klinische Pfade durchgeführt werden.

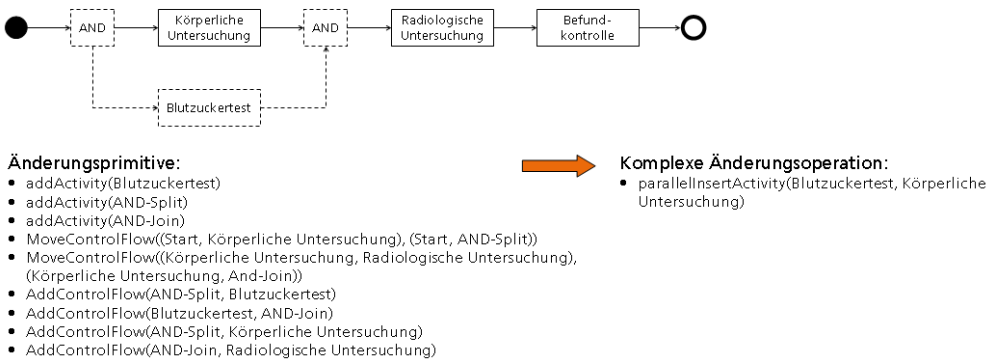
Die Auswahl und Anwendung der Änderungsprimitive zur Anpassung des Prozesses ist jedoch sehr aufwändig und erfordert das Verständnis der zugrunde liegenden technischen Systematik. Das medizinische Personal im Krankenhaus hat dazu kaum die Zeit. Effizienzsteigerungen durch die gleichzeitige Durchführung derselben Änderung an unterschiedlichen Stellen im Prozess sind nicht möglich. Auch können Anwender durch die Spezifikation semantischer Zusammenhänge zwischen Abweichungsmaßnahmen nicht in ihrer Arbeit unterstützt werden. Die Ausführbarkeit des Prozesses nach einer Anpassung ist nicht garantiert. Fehler im Kontrollfluss werden zwar automatisch erkannt, müssen aber von den Anwendern ausgeräumt werden; die Integrität des Datenflusses wird nicht adressiert.

3.2.6 Nutzung komplexer Änderungsoperationen

Bei ADEPT2 (Application Development Based on Pre-modeled Encapsulated Process Templates) handelt es sich um ein adaptives WfMS, das ebenfalls strukturelle Ad-hoc-Prozessänderungen zur Laufzeit unterstützt [Reichert & Dadam 1998, Reichert et al. 2005]. Im Gegensatz zu CAKE2 und WASA2 bietet ADEPT2 Anwendern jedoch keine Änderungsprimitive zur Adaption von Prozessinstanzen an, sondern realisiert Operationen auf einem semantisch hohen Niveau; z.B. entspricht das Einfügen von Aktivitäten parallel zu einer Menge bereits existierender Prozessaktivitäten einer solchen komplexen Änderungsoperation. Diese komplexen Änderungsoperationen realisieren die von Weber et al. definierten »Change Pattern« [Weber et al. 2008]. Die nachfolgende Abbildung stellt auf Basis des obigen Beispiels dar, auf welche Weise komplexe Änderungsoperationen die Abstraktion von Details einer Prozessänderung ermöglichen. Während insgesamt neun einfache Änderungsoperationen not-

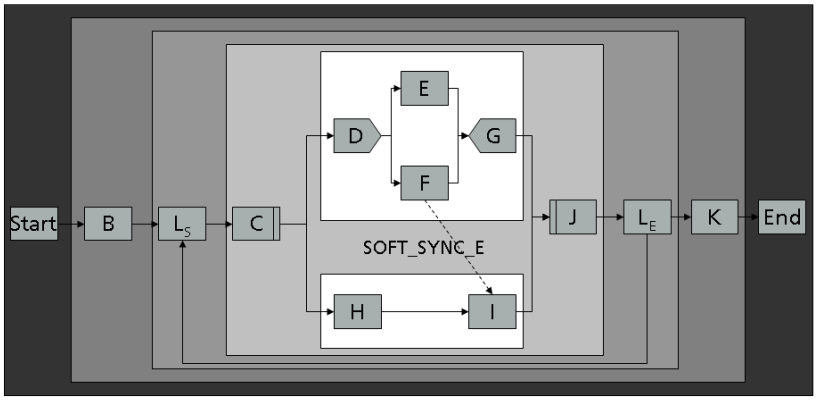
wendig sind, um den Blutzuckertest zusätzlich zum Standardlabor einzufügen, wird dafür in ADEPT2 nur eine einzige komplexe Änderungsoperation benötigt.

Abbildung 14 Abstraktion von Details einer Prozessänderung durch Nutzung von komplexen Änderungsoperationen



Im Gegensatz zu Änderungsprimitiven ist es bei komplexen Änderungsoperationen möglich, Vor- und Nachbedingungen zu definieren, die gelten müssen, damit eine Operation auf eine Prozessdefinition bzw. –instanz angewendet werden kann. Bevor ADEPT2 die Ausführung einer Modifikation am Prozess erlaubt, müssen die Elemente im Prozessgraph markiert sein, die von der Änderung betroffen sind. Liegt keine eindeutige oder eine unzulässige Markierung vor, kann die Operation nicht verwendet werden. Auf diese Weise schließt ADEPT2 bereits im Vorfeld Prozessänderungen aus, die zu inkonsistenten Zuständen wie Verklemmungen führen können, und bereitet damit die Basis für die Ausführbarkeit von Prozessdefinitionen. Grundlage für die effiziente Überprüfung der korrekten Strukturierung von Prozessdefinitionen ist das formale block-orientierte Wf-Metamodell von ADEPT2. Die Blockstruktur erzwingt, dass Sequenzen von Aktivitäten, parallele und bedingte Verzweigungen sowie Schleifenkonstrukte als symmetrische Blöcke spezifiziert werden, die ineinander geschachtelt sein können, sich aber nicht überlappen dürfen (siehe Abbildung 15). Bei einer Anpassung muss demnach nur die Korrektheit des betroffenen Blockes kontrolliert werden; ist jeder Block innerhalb einer Prozessdefinition korrekt, so gilt dies auch für den gesamten Prozess.

Abbildung 15 Prinzip der Blockstrukturierung in ADEPT2 [Reichert 2000b]



Eine Ausnahme von der Blockstruktur bilden so genannte Synchronisationskanten, die es ermöglichen, Ausführungsabhängigkeiten zwischen Aktivitäten zweier paralleler Zweige festzu-

legen; mit Hilfe entsprechender Analyseverfahren schließt ADEPT2 Situationen aus, in denen es aufgrund von unzulässigen Anordnungen von Synchronisationskanten zu Verklemmungen kommen kann. Datenflüsse werden in ADEPT2 explizit festgelegt. Im Gegensatz zu vielen anderen adaptiven WfMS stellt ADEPT2 nach einer Prozessänderung nicht nur die Korrektheit des Kontrollflusses sicher, sondern prüft nach dem Löschen oder Verschieben von Prozessaktivitäten automatisch, ob alle Anwendungskomponenten immer noch mit denen von ihnen benötigten Datenobjekten versorgt sind. Ist dies nicht der Fall, generiert das System alternative Vorschläge, wie das Problem behoben werden kann [Reichert 2000b]. Die nachfolgende Tabelle gibt zunächst einen Überblick über die Analyseergebnisse.

Tabelle 11 Ergebnisse der Analyse von ADEPT2 zur Unterstützung manueller Abweichungsmaßnahmen

Zentrale Anforderung		**Resultierende Teilanforderung**		**ADEPT2**
2.	Unterstützung manueller Abweichungsmaßnahmen	2.1	Manuelle Abweichungsmaßnahmen vom klinischen Pfad müssen für Fachanwender durchführbar sein (siehe Kapitel 2.2.3.1, 2.2.3.4.1, 2.2.3.4.4)	**Nein**
		2.2	Fachanwender und WfMS müssen klar zwischen Standardprozess und vom Standard abweichenden Maßnahmen unterscheiden können (siehe Kapitel 2.2.3.1, 2.2.3.2)	**Ja**
		2.3	Die Wahl der Abweichungsmaßnahmen muss in Abhängigkeit des Behandlungsfalls variieren können (siehe Kapitel 2.2.3.3)	**Ja**
		2.4	Fachexperten müssen sowohl die Möglichkeit haben sofort als auch verzögert auf eine Varianz zu reagieren (siehe Kapitel 2.2.3.4.2)	**Ja**
		2.5	Abweichungsmaßnahmen können zu unterschiedlichen Zeitpunkten und an unterschiedlichen Stellen im Prozess wiederholt werden (siehe Kapitel 2.2.3.4.3)	**Ja**
		2.6	Dieselben Abweichungsmaßnahmen lassen sich gleichzeitig an mehreren Stellen des Prozesses durchführen (siehe Kapitel 2.2.3.4.5)	**Nein**
		2.7	Semantische Abhängigkeiten zwischen Abweichungsmaßnahmen müssen bei ihrer Auswahl berücksichtigt werden (siehe Kapitel 2.2.3.4.6)	**Nein**
		2.8	Dieselben Abweichungsmaßnahmen können unabhängig von bestimmten klinischen Pfaden durchgeführt werden (siehe Kapitel 2.2.3.4.7)	**Ja**
		2.9	Die Ausführung des klinischen Pfades muss auch nach der Durchführung manueller Abweichungsmaßnahmen möglich sein (siehe Kapitel 2.1.5)	**Ja** (mit Einschränkungen)

Obwohl komplexe Änderungsoperationen eine Menge von Änderungsprimitiven kapseln und damit die Durchführung von Anpassungen an Prozessdefinitionen und Instanzen vereinfachen, sind an ihre Nutzung immer noch fundierte technische Kenntnisse geknüpft. Z. B. müssen Anwender nach den Regeln von ADEPT2 die richtigen Prozessaktivitäten markieren, bevor die gewünschte Änderungsoperation überhaupt frei geschaltet wird. Damit entsprechen die Vor- und Nachteile im Hinblick auf die Unterstützung manueller Abweichungsmaßnahmen

denen von CAKE2 und WASA2 aus Kapitel 3.2.5 mit einer Ausnahme: Prozessdefinitionen, die in ADEPT2 angepasst wurden, können weiterhin fehlerfrei ausgeführt werden. Durch die Nutzung von komplexen Änderungsoperationen können Fehler im Kontrollfluss ausgeschlossen werden. Aufgrund der expliziten Bekanntgabe der Datenflüsse und der Bereitstellung entsprechender Analysemethoden werden auch hier Inkonsistenzen erkannt; diese müssen von den Anwender allerdings manuell behoben werden. Beim Einfügen neuer Aktivitäten müssen diese außerdem durch die Zuordnung von Anwendungskomponenten operationalisiert werden, bevor die Ausführung des Prozesses wieder aufgenommen werden kann.

3.3 Ansätze zur Unterstützung von Prozessvarianten

Strategien zur ad hoc Adaption von Prozessen wie in Kapitel 3.2.5 und 3.2.6 beschrieben, ermöglichen maximale Flexibilität. Allerdings sind sie auch mit dem höchsten Aufwand für das medizinische Personal verbunden und erfordern üblicherweise ein tiefgreifendes technisches Verständnis. Darüber hinaus ist mit steigender Komplexität der Abweichungsmaßnahmen die Wahrscheinlichkeit für das Auftreten fachlicher Fehler bei ad hoc Änderungen groß, da es bei professions- und stationsübergreifenden Prozessen oftmals schwierig ist, alle Auswirkungen der Modifikation im Voraus abzuschätzen. Für Varianz, deren Auswirkungen zur Modellierungszeit weitgehend bekannt sind, können Prozessvarianten spezifiziert werden, deren Auswahl dafür sorgt, dass statt dem vom klinischen Pfad vorgegebenen Standardablauf ein alternatives Vorgehen gewählt wird. Insbesondere bei komplexen Abweichungen vom klinischen Pfad, die aus vielen einzelnen Maßnahmen bestehen, können Prozessvarianten erheblich zur Reduktion des Arbeitsaufwandes der Anwender beitragen. Außerdem ermöglichen sie die Definition eines Standardvorgehens auch im Fall einer Varianz. Prozessvarianten als mögliche Alternativen zum Standardprozess können auf sehr unterschiedliche Weise umgesetzt werden. Allen Ansätzen ist gemein, dass sie als Voraussetzung für die Spezifikation von Prozessvarianten singuläre Abweichungsmaßnahmen unterstützen, auch wenn sie nicht alle der in Kapitel 2.1.5 beschriebenen Anforderungen an deren Durchführung erfüllen.

3.3.1 Flexibilität durch Realisierung von Workflow Patterns

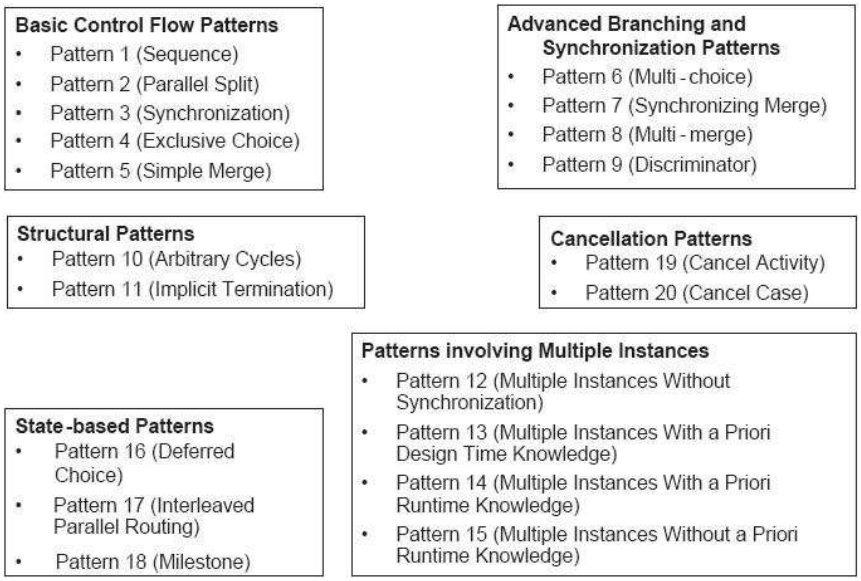
YAWL (Yet Another Workflow Language) ist ein Akronym für eine Prozessmodellierungssprache basierend auf gefärbten Petrinetzen [van der Aalst & ter Hofstede 2002, van der Aalst & ter Hofstede 2005] und gleichzeitig die Bezeichnung für ein flexibles WfMS [van der Aalst et al. 2004, ter Hofstede et al. 2010]. Die Nutzung von Petri-Netzen wird motiviert durch die formale Beweisbarkeit der strukturellen Korrektheit [van der Aalst 1997], die zustandbasierte Ausführungssemantik und die Fülle an vorhandenen Analysemethoden. Die Entwicklung der Modellierungssprache YAWL fußt auf der Identifikation von Mustern bei der Erstellung von Prozessdefinitionen, den so genannten »Workflow Patterns«. Riehle & Züllighoven definieren Muster als

> »abstraction from a concrete form which keeps recurring in specific nonarbitrary contexts.« [Riehle & Züllighoven 1996]

Solche Muster ermöglichen also sowohl die Abstraktion von einer konkreten zugrunde liegenden Technologie als auch von einer bestimmten Domäne und haben stattdessen einen Allgemeingültigkeitsanspruch; dies gilt in analoger Weise auch für die in [Weber et al. 2008] vorgestellten Change Pattern. Die nachfolgende Abbildung gibt einen Überblick über die im Rahmen

des verhaltensorientierten Aspekts identifizierten Workflow Pattern, die auch als »Control Flow Pattern« bezeichnet werden [Russell et al. 2006].

Abbildung 16 Überblick über Control Flow Pattern [Van der Aalst et al. 2003][30]



Während z. B. über das »Sequence« Pattern bereits zum Zeitpunkt der Erstellung der Prozessdefinition eine fixe Reihenfolge vorgegeben wird, in der die Aktivitäten ausgeführt werden, ermöglicht das Pattern »Interleaved Parallel Routing« die Festlegung dieser Reihenfolge zur Laufzeit. D. h. die Anwender können z. B. fall- und situationsbezogen bestimmen, ob die Bild gebende Diagnostik vor oder nach der Entnahme von Laborproben erfolgen soll; die Reihenfolge kann damit von Parametern abhängen, die nicht zum Zeitpunkt der Modellierung, sondern erst zur Ausführung bekannt sind, wie z. B. die aktuelle Belegung der Ressourcen. Ähnlich wie bei Late Bindung und Late Modeling (Kapitel 3.2.2 und 3.2.3) lassen sich auf diese weise manuelle Abweichungsmaßnahmen in vordefinierten Regionen umsetzen.

Neben den Control Flow Pattern sind weitere Pattern definiert, die sich auf andere Workflow Aspekte beziehen, nämlich die informationsbezogene [Russell et al. 2004a] und die organisatorische Perspektive [Russell et al. 2004b]. Es wird deutlich, dass die Prozessausführung umso flexibler wird, je mehr solcher Workflow Pattern ein WfMS unterstützt. Um die Fähigkeit eines WfMS bezüglich seines Umgangs mit Ausnahmefällen bewerten zu können, stellen Russell et al. zusätzlich Pattern zur Behandlung von Ausnahmeereignissen vor [Russel et al. 2006]. Es werden mögliche Typen von Ausnahmefällen beschrieben und Mechanismen zum Umgang mit solchen Situationen identifiziert. Die Notation von Ausnahmeereignissen, die z. B. zur wiederholten Durchführung einer fehlgeschlagenen Aktivität, zur Reallokation von Ressourcen oder zu Kompensationsvorgängen führen, hat aber nichts mit strukturellen Ände-

[30] Siehe http://www.workflowpatterns.com/ für eine ausführliche Beschreibung der einzelnen Workflow Patterns

rungen an Prozessdefinitionen und -instanzen zu tun, wie sie durch Varianz von klinischen Pfaden notwendig werden können.

Prinzipiell müssen WfMS, die auf die Realisierung von Workflow Pattern setzen, um die Prozessausführung zu flexibilisieren, ein mächtiges Wf-Metamodell bereitstellen, das eine Vielzahl von Konstrukten anbietet, die genutzt werden können, um Prozesse zu variieren. Da sich mit Workflow Pattern prinzipiell auch komplexe Abweichungen von Standardprozessen realisieren lassen, können sie der Umsetzung von Prozessvarianten im Kontext klinischer Pfade dienen. Tabelle 12 stellt die Analyseergebnisse dar.

Tabelle 12 Ergebnisse der Analyse von YAWL zur Unterstützung von Prozessvarianten

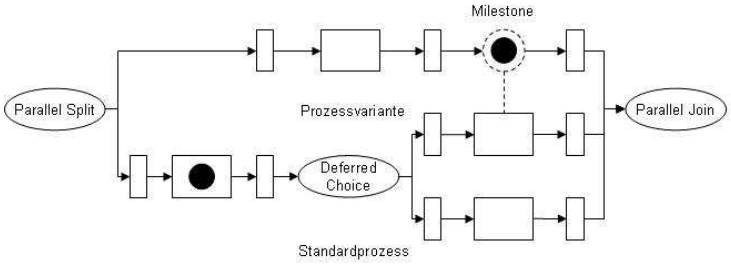
Zentrale Anforderung	**Resultierende Teilanforderung**		**YAWL**
3. Unterstützung von Prozessvarianten	3.1	Prozessvarianten als standardisierte Bündel von Abweichungsmaßnahmen müssen für Fachanwender modellierbar sein (siehe Kapitel 2.2.3.4.1, 2.2.3.4.4)	**Nein**
	3.2	Fachanwender und WfMS müssen klar zwischen Standardprozess und Variante unterscheiden können (siehe Kapitel 2.2.3.1, 2.2.3.2)	**Nein**
	3.3	Semantische Abhängigkeiten zwischen einzelnen Abweichungsmaßnahmen einer Prozessvariante müssen berücksichtigt werden (siehe Kapitel 2.2.3.4.6)	**Ja** (hohe Komplexität)
	3.4	Fachanwender müssen Prozessvarianten in Abhängigkeit des Behandlungsfalls anpassen können (siehe Kapitel 2.2.3.4.1, 2.2.3.4.4)	**Ja** (hohe Komplexität)
	3.5	Prozessvarianten können unabhängig von bestimmten klinischen Pfaden definiert und eingesetzt werden (siehe Kapitel 2.2.3.4.7)	**Nein**
	3.6	Konflikte mit manuellen Abweichungsmaßnahmen werden vermieden (siehe Kapitel 2.2.3.4.1)	**Ja** (bei Worklets)
	3.7	Die Ausführung des klinischen Pfades muss auch nach der Auswahl einer Prozessvariante möglich sein (siehe Kapitel 2.1.5)	**Ja**

Um eine Prozessvariante explizit als alternativen Weg der Prozessausführung zu beschreiben, kann in YAWL wie in vielen anderen Prozessmodellierungssprachen das Konstrukt »Exclusive Choice« verwendet werden. Dieses Konstrukt hat jedoch mehrere Nachteile. So ist es damit nicht möglich, zwischen dem Standardvorgehen und einer Varianz zu unterscheiden. Außerdem ist nicht vorgesehen, dass eine komplexe Variante mit Auswirkungen auf unterschiedliche Prozessregionen als logische Einheit spezifiziert und ausgewählt werden kann. Die einfachste Methode eine über die Prozessdefinition verteilte Variante als logische Einheit zu spezifizieren, besteht in der Einbettung großer Teile des Standardprozesses in die alternativen Zweige. Dies macht die Modellierung von Prozessen jedoch extrem komplex, reduziert die Verständlichkeit der Prozessdefinitionen und maximiert den Pflegeaufwand.

Während Fachanwender die Entscheidung für oder gegen eine Variante mit »Exclusive Choice« explizit treffen, ermöglicht das Konstrukt »Deferred Choice« dem System die Entscheidung implizit mitzuteilen. Die Wahl des alternativen Zweiges hängt hier davon ab, welche

Option zuerst selektiert wurde. Durch eine Kombination von »Deferred Choice« mit dem Konstrukt »Milestone« kann die Prozessdefinition so gestaltet werden, dass der Zeitpunkt, an dem sich Anwender für den Standardprozess oder die Variante aktiv entscheiden können, von dem Fortschreiten der Prozessausführung in anderen Teilzweigen abhängt. Die folgende Abbildung zeigt ein Petri-Netz mit der Markierung, in der es möglich ist, die Prozessvariante alternativ zum Standardzweig zu wählen. Ist der »Milestone« nicht besetzt ist, steht die Prozessvariante auch nicht zur Auswahl. Es ist jedoch fraglich, ob dieses Konstrukt in realen klinischen Anwendungsszenarios standardmäßig zum Umgang mit Varianz eingesetzt werden kann.

Abbildung 17 Nutzung von Meilensteinen zur Auswahl von Prozessvarianten



Semantische Abhängigkeiten und Variabilität innerhalb der Prozessvariante selbst können durch die Realisierung der vorgestellten Workflow Pattern berücksichtigt werden, jedoch auf Kosten hoher Komplexität der resultierenden Prozessdefinitionen. Das in Kapitel 3.2.2 vorgestellt Worklet-Konzept wurde mit YAWL kombiniert, so dass die Unterstützung manueller Abweichungsmaßnahmen durch den Late Binding Mechanismus konfliktfrei möglich ist. Da alle Varianten Bestandteil derselben Prozessdefinition sind und den Korrektheitskriterien der Modellierungssprache unterliegen, ist ihre Ausführung prinzipiell möglich. Die Einbettung der von Anwendern neu definierten Worklets kann jedoch Probleme bereiten.

Mit der Menge an unterschiedlichen Workflow Pattern, die unterstützt werden, nimmt die Verständlichkeit der Prozessdefinition für Fachexperten natürlich ab. Die Petri Netz-basierte Modellierungssprache YAWL ist ohnehin nicht für die Nutzung von Ärzten oder Krankenschwestern konzipiert worden; dementsprechend hat das medizinische Personal auch nicht die Möglichkeit, selbständig Varianten zu erstellen. Bei der Nutzung des Workflow Pattern »Interleaved Routing«, bei vielen alternativen Verzweigungen und redundanten Prozessabschnitten, kann außerdem der Unterschied zwischen Standard und Varianz schnell verwischen. Prozessvarianten sind bei YAWL immer vollständig in die Prozessdefinition integriert und können auch im Fall von prozess-übergreifender Varianz nicht unabhängig davon spezifiziert werden.

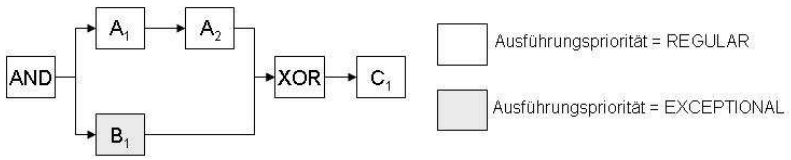
3.3.2 Festlegung von Ausführungsprioritäten

Das adaptive WfMS ADEPT2 wurde bereits in Kapitel 3.2.6 vorgestellt. Neben komplexen Änderungsoperationen zur dynamischen Adaption von Prozessen zur Laufzeit stellt es auch Möglichkeiten zum Umgang mit planbaren Abweichungen bereit [Reichert et al. 2002]. Dem Ansatz liegt die Motivation zur Unterstützung bekannter Abweichungen von Standardprozessen zugrunde, wobei anders als bei YAWL explizit die Unterscheidbarkeit zwischen Standard und Varianz gefordert wird. Das Wf-Metamodell von ADEPT2 ist wie Petri-Netze formal fundiert, verfügt über eine zustandsbasierte Ausführungssemantik, eine graphische Repräsen-

tation und erzwingt aufgrund seiner Blockstruktur die Modellierung strukturell korrekter Prozessdefinitionen; dabei wird auch die Integrität des Datenflusses überprüft (siehe Kapitel 3.2.6).

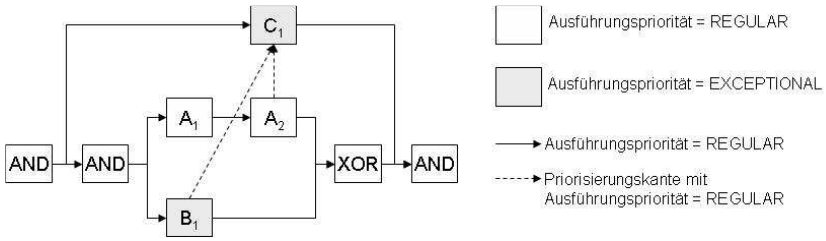
Dieses Wf-Metamodell wurde nun um die Möglichkeit zur Definition von Ausführungsprioritäten erweitert, um eine Abweichung vom Standardverlauf in Folge einer Varianz kenntlich zu machen [Reichert et al. 2002]. Zu diesem Zweck wird den Aktivitäten der Prozessdefinition ein zusätzliches Attribut zugeordnet, dem, falls es sich um eine Ausnahmeaktivität handelt, der Wert »EXCEPTIONAL« zugewiesen wird; normale Aktivitäten erhalten den Wert »REGULAR«. Bevorzugte Ausführungszweige können modelliert werden, indem ausgehend von einer parallelen Verzweigung (Workflow Pattern »Parallel Split« bzw. »AND«) der einen nachfolgenden Aktivität der Wert »REGULAR«, der anderen nachfolgenden Parallelaktivität der Wert »EXCEPTIONAL« zugewiesen wird. Die Synchronisation erfolgt gemäß Workflow Pattern »Discriminator« bzw. »XOR«, so dass nur einer der parallelen Zweige ausgeführt werden muss, um mit der weiteren Behandlung fortzufahren. Die nachfolgende Abbildung illustriert das Konzept.

Abbildung 18 Festlegung von Ausführungsprioritäten zur Signalisierung von Abweichungen vom Standardablauf



Durch die Modellierung von Ausführungsprioritäten lassen sich auch planbare Vorwärts- und Rückwärtssprünge umsetzen. Bei Vorwärtssprüngen werden parallel zu dem normalerweise auszuführenden Prozessteil Ausnahmeaktivitäten eingefügt, deren Auswahl dazu führt, dass Teile des Standardprozesses übersprungen werden. In ADEPT2 ist es auch möglich, übersprungene Aktivitäten nachzuholen. Wie in der folgenden Abbildung dargestellt, handelt es sich bei C_1 um eine Ausnahmeaktivität, deren Bearbeitung jedoch vorgezogen werden kann. Die parallele Synchronisation am Ende des Blockes stellt dabei sicher, dass die Parallelaktivitäten nachgeholt werden, bevor die Ausführung fortschreitet. Um zu signalisieren, dass es sich bei C_1 nur dann um eine Ausnahmeaktivität handelt, wenn sie vor A_2 (bzw. B_1) ausgeführt wird, wird zwischen A_2 (bzw. B_1) und C_1 eine Priorisierungskante mit dem Wert »REGULAR« eingefügt; d. h. falls die Bearbeitung von C_1 nicht vorgezogen wurde, wird C_1 nach Abschluss von A_2 oder B_1 als regulärer Arbeitsschritt angeboten.

Abbildung 19 Planbarer Vorwärtssprung mit Nachholen von übersprungenen Aktivitäten



Mit Hilfe der Konstrukte Ausführungsprioritäten und Priorisierungskanten lassen sich in ADEPT2 Varianten vom Standardprozess modellieren. Im Gegensatz zu YAWL ist es sowohl für Anwender als auch das WfMS möglich, zwischen Standard und Varianz zu unterscheiden. Analog zu YAWL lassen sich auch semantische Abhängigkeiten zwischen einzelnen Abweichungsmaßnahmen definieren oder Prozessvarianten variabel gestalten, allerdings ebenfalls auf Kosten hoher Komplexität. Durch die automatische Sicherstellung der Korrektheit des Kontrollflusses sowie der Vollständigkeit des Datenflusses kann die Ausführbarkeit von Prozessen in ADEPT2 zu allen Zeitpunkten gewährleistet werden. Die Ergebnisse der Analyse sind in der nachfolgenden Tabelle aufgelistet.

Tabelle 13 Ergebnisse der Analyse von ADEPT2 zur Unterstützung von Prozessvarianten

Zentrale Anforderung	**Resultierende Teilanforderung**		**ADEPT2**
3. Unterstützung von Prozessvarianten	3.1	Prozessvarianten als standardisierte Bündel von Abweichungsmaßnahmen müssen für Fachanwender modellierbar sein (siehe Kapitel 2.2.3.4.1, 2.2.3.4.4)	**Nein**
	3.2	Fachanwender und WfMS müssen klar zwischen Standardprozess und Variante unterscheiden können (siehe Kapitel 2.2.3.1, 2.2.3.2)	**Ja**
	3.3	Semantische Abhängigkeiten zwischen einzelnen Abweichungsmaßnahmen einer Prozessvariante müssen berücksichtigt werden (siehe Kapitel 2.2.3.4.6)	**Ja** (hohe Komplexität)
	3.4	Fachanwender müssen Prozessvarianten in Abhängigkeit des Behandlungsfalls anpassen können (siehe Kapitel 2.2.3.4.1, 2.2.3.4.4)	**Ja** (hohe Komplexität)
	3.5	Prozessvarianten können unabhängig von bestimmten klinischen Pfaden definiert und eingesetzt werden (siehe Kapitel 2.2.3.4.7)	**Nein**
	3.6	Konflikte mit manuellen Abweichungsmaßnahmen werden vermieden (siehe Kapitel 2.2.3.4.1)	**Nein**
	3.7	Die Ausführung des klinischen Pfades muss auch nach der Auswahl einer Prozessvariante möglich sein (siehe Kapitel 2.1.5)	**Ja**

Da alle Prozessvarianten Bestandteil des klinischen Pfades sind und auf dieselbe Modellierungsmethodik zurückgehen, ist es für das medizinische Personal ohne technische Kenntnisse nicht möglich, Varianten eigenständig zu erstellen. Ebenso lassen sich variable Abläufe nicht unabhängig von den einzelnen Pfaden modellieren; auch wenn die Maßnahmen selbst in Form von Subprozessen modular zur Verfügung gestellt werden, müssen die bedingten Verzweigungsknoten und die Ausnahmekanten für jeden Pfad separat definiert werden. Da ADEPT2 zusätzlich ad hoc Änderungen an Prozessdefinitionen und –instanzen auf Basis komplexer Änderungsoperationen erlaubt, ist eine Kombination von Prozessvarianten und manuellen Änderungsmaßnahmen prinzipiell möglich, aber nicht immer konfliktfrei zu bewerkstelligen. Z. B. kann es passieren, dass eine manuelle Anpassung in Folge der nachträglichen Auswahl eines Ausnahmezweiges übersprungen wird und gegebenenfalls im Ausnahmezweig wiederholt durchgeführt werden muss.

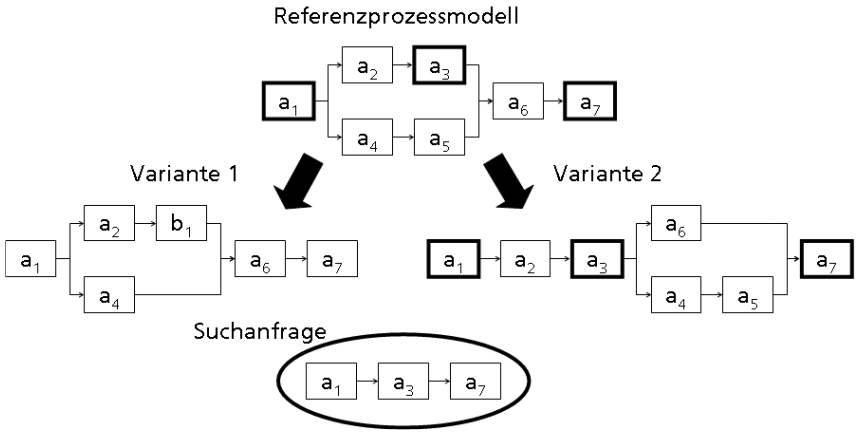
3.3.3 Konfiguration von Prozessdefinitionen

Eine potentielle Antwort auf die Probleme der zunehmenden Komplexität von Prozessdefinitionen und den verteilten Auswirkungen von Varianz gibt der »Multi-Model« Ansatz, bei dem für jede mögliche Varianz eine eigene Prozessdefinition als Prozessvariante modelliert und verwaltet wird. Prozessvarianten repräsentieren in diesem Fall also Variationen von einem Hauptprozess, dem Referenzprozessmodell [Schütte 1997, Fettke & Loos 2003, Fettke et al. 2006]. Gemäß Scheer handelt es sich bei einem Referenzmodell um ein Modell,

> »das als Ausgangspunkt für die Entwicklung auf konkrete Aufgabenstellungen bezogener Problemlösungen dienen kann« [Scheer 1998][31].

Der Standardablauf, beschrieben durch den klinischen Pfad, entspricht demzufolge dem Referenzprozessmodell, von dem es, je nach konkreter Aufgabenstellung in Folge einer Varianz, mehrere Prozessvarianten geben kann. Lu & Sadiq beschreiben z. B. einen Multi-Modell Ansatz, bei dem alternative Prozessdefinitionen aus manuellen Änderungen an der ursprünglichen Definition hervorgehen und in einem entsprechenden Varianten-Repository (PVR) hinterlegt werden [Lu & Sadiq 2006]. Soll der Prozess instanziiert werden, kann dies entweder auf Basis der Original-Prozessdefinition oder einer ihrer Varianten geschehen. Die Suche nach passenden Varianten erfolgt an Hand strukturierter Suchanfragen; diese Anfragen entsprechen Kombinationen von Elementen der Prozessdefinition bzw. Teilstrukturen, die mit den im Repository verwalteten Prozessvarianten abgeglichen werden können. So ist es möglich, je nach Fall die passende Prozessdefinition für die Instanziierung und Ausführung zu selektieren. Die folgende Abbildung illustriert das Verfahren beim Multi-Modell Ansatz.

Abbildung 20 Ableitung von Prozessvarianten vom Referenzprozessmodell und Suche an Hand von Teilprozessstrukturen



Vom Referenzprozessmodell existieren zwei Varianten, wobei Aktivität a_3 in Variante 1 durch b_1 ersetzt wurde, währenddessen in Variante 2 lediglich die Reihenfolge der Prozessausführung variiert wurde. Der Suchanfrage zu Folge, sollen nur solche Prozessdefinitionen in Be-

[31] Vgl. Scheer 1998, S. 6

tracht gezogen werden, bei denen a_1, a_3 und a_7 in sequentieller Reihenfolge ausgeführt werden; dies trifft auf das Referenzprozessmodell und Variante 2 zu.

Ein gravierender Nachteil des Multi-Modell Ansatzes ist der hohe Pflegeaufwand, der mit steigender Anzahl von Varianten im Varianten-Repository weiter zunimmt [Hallerbach et al. 2008b]. Darüber hinaus erhöht sich die Menge der pro klinischen Pfad zu verwaltenden Prozessdefinitionen enorm, wenn auch Kombinationen aus verschiedenen Varianzen berücksichtigt werden müssen (z. B. Diabetes mellitus und Hypertonie). Aus diesen Gründen werden Lösungsansätze entwickelt, die die Einbettung der Prozessvarianten in das Referenzprozessmodell im Sinne eines »Single-Model Ansatzes« erlauben, so dass bei Bedarf nur eine einzige Prozessdefinition angepasst werden muss. In der Literatur zu Softwareprodukten wird die Konfiguration von Produkten als ein komfortables Verfahren zur Abbildung von Produktfamilien durch die Integration von Varianten in einem gemeinsamen Modell beschrieben [Bachmann & Bass 2001, Becker et al. 2001, Halmans & Pohl 2003, Svahnberg et al. 2005]. Puhlmann et al. schildern im Rahmen des PESOA (Process Family Engineering in Service-Oriented Architectures) Projektes die Übertragbarkeit des Ansatzes auf Prozessdefinitionen und ihre Varianten [Puhlmann et al. 2005, Bayer et al. 2005]. Das Wf-Metamodell wird durch die Deklaration von Variationspunkten erweitert; diese Variationspunkte ermöglichen die Auswahl von Varianten an definierten Stellen in der Prozessdefinition.

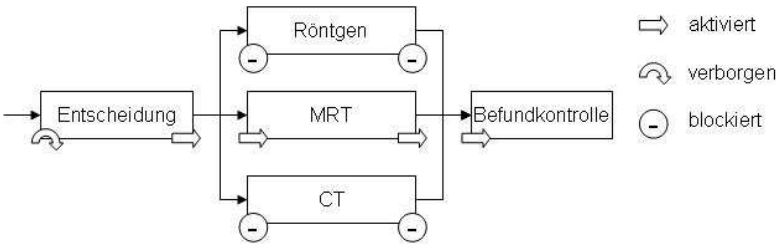
Gottschalk et al. stellen ein generisches Konzept für die Konfiguration von Prozessdefinitionen vor [Gottschalk et al. 2008], das u. a. am Beispiel von Erweiterungen an EPCs (Event Process Chains) und YAWL konkret demonstriert wird; die so ergänzten Modellierungssprachen werden entsprechend als C-EPC (Configurable Process Chains) [Rosemann & van der Aalst 2007, La Rosa 2009[32]] und C-YAWL (Configurable YAWL) [Gottschalk et al. 2008] bezeichnet. Der konfigurierbare Prozess besteht zum einen aus einem Basisprozess, der das kleinste gemeinsame Vielfache aller potentiellen Prozessvarianten repräsentiert, und einer Standardkonfiguration. Diese Standardkonfiguration wird benötigt, da der Basisprozess nicht nur den Standardablauf, sondern auch sämtliche Prozessvarianten umfasst, und in dieser Form gegebenenfalls gar nicht ausgeführt werden kann; ohne Standardkonfiguration könnten z. B. Aktivitäten bearbeitet werden, die zueinander in Konflikt stehen. Abwandlungen der Standardkonfiguration resultieren in einer Prozessvariante.

Die Konfiguration erfolgt in Abhängigkeit der Belegung von Ports, die Prozessaktivitäten mit den eingehenden (»inflow ports«) und ausgehenden (»outflow ports«) Kontrollflusskanten verbinden. Je nach Belegung der Ports werden bestimmte Bereiche in der Prozessdefinition als aktiviert, verborgen oder blockiert konfiguriert. Wird ein Port aktiviert, so erfolgt die Durchführung der zugehörigen Aktivität in der üblichen Art und Weise. Ist ein Port hingegen verborgen, wird die Ausführung der Aktivität übersprungen. Blockaden können im Anschluss an eine parallele oder bedingte Verzweigung ausgelöst werden und haben zur Folge, dass die entsprechenden Zweige oder eine bestimmte Kombination von Zweigen nicht ausgeführt werden kann. Die folgende Abbildung zeigt die Konfiguration eines Prozessabschnittes zur Bild gebenden Diagnostik im Fall einer Schwangerschaft; da Röntgenuntersuchungen und auch CTs (aufgrund ihrer ionisierenden Strahlung) bei schwangeren Patientinnen nicht durchgeführt

[32] Vgl. La Rosa 2009, S. 52-76

werden sollten, ist die Entscheidungsaktivität verborgen, die Aktivitäten »Röntgen« und »CT« blockiert und nur die Aktivität »MRT« aktiviert.

Abbildung 21 Konfiguration eines Prozessabschnittes zur Bild gebenden Diagnostik bei Vorliegen einer Schwangerschaft



Durch die Konfiguration von Prozessdefinitionen ist es möglich, eine Varianz vor der Instanziierung eines klinischen Pfades so zu berücksichtigen, dass die Anwender keine weiteren Entscheidungen mehr im Hinblick auf diese Varianz zur Laufzeit treffen müssen. Konfigurierte Prozessdefinitionen können außerdem übersichtlicher sein als Definitionen, die sämtliche alternativen Ausführungszweige noch zur Laufzeit umfassen. Die Vorteile der Konfiguration können jedoch nur dann genutzt werden, wenn die entsprechende Varianz bereits zum Zeitpunkt der Instanziierung des klinischen Pfades bekannt ist. Dies setzt der Anwendbarkeit des Konzeptes im klinischen Bereich enge Grenzen. Um auch auf Varianz reagieren zu können, die erst zur Laufzeit entsteht, können bedingte Verzweigungen genutzt werden, die zu den in Kapitel 3.3.1 geschilderten Problemen führen. Tabelle 14 fasst die Ergebnisse der Analyse zusammen.

Tabelle 14 Ergebnisse der Analyse von C-YAWL und C-EPC zur Unterstützung von Prozessvarianten

Zentrale Anforderung		**Resultierende Teilanforderung**		**C-YAWL/C-EPC**
3.	Unterstützung von Prozessvarianten	3.1	Prozessvarianten als standardisierte Bündel von Abweichungsmaßnahmen müssen für Fachanwender modellierbar sein (siehe Kapitel 2.2.3.4.1, 2.2.3.4.4)	**Nein**
		3.2	Fachanwender und WfMS müssen klar zwischen Standardprozess und Variante unterscheiden können (siehe Kapitel 2.2.3.1, 2.2.3.2)	**Ja**
		3.3	Semantische Abhängigkeiten zwischen einzelnen Abweichungsmaßnahmen einer Prozessvariante müssen berücksichtigt werden (siehe Kapitel 2.2.3.4.6)	**Ja**
		3.4	Fachanwender müssen Prozessvarianten in Abhängigkeit des Behandlungsfalls anpassen können (siehe Kapitel 2.2.3.4.1, 2.2.3.4.4)	**Nein**
		3.5	Prozessvarianten können unabhängig von bestimmten klinischen Pfaden definiert und eingesetzt werden (siehe Kapitel 2.2.3.4.7)	**Nein**
		3.6	Konflikte mit manuellen Abweichungsmaßnahmen werden vermieden (siehe Kapitel 2.2.3.4.1)	**Nein** (ggf. Worklets in C-YAWL)

3.7	Die Ausführung des klinischen Pfades muss auch nach der Auswahl einer Prozessvariante möglich sein (siehe Kapitel 2.1.5)	**Ja** (bei C-YAWL)

Durch die Vorgabe einer Basiskonfiguration ist die Unterscheidung zwischen Standard und Variante prinzipiell möglich; sobald sich ein Anwender für eine andere Konfiguration entscheidet, weicht er automatisch vom klinischen Pfad ab. Worin die aktuellen Abweichungsmaßnahmen bestehen, kann allerdings nur im direkten Vergleich mit der Prozessdefinition ermittelt werden, die durch Anwendung der Basiskonfiguration entsteht. Es ist möglich, semantische Abhängigkeiten zwischen konfigurierbaren Aktivitäten zu definieren; wirken sich zwei Konfigurationen auf denselben Port einer Aktivität aus, so dominieren die Einstellungen der ersten Konfiguration. Mit einer ansteigenden Zahl von Varianten nimmt die Komplexität allerdings rasant zu. Da die Korrektheit aller Konfigurationen zur Modellierungszeit getestet werden kann, kann die Ausführbarkeit nach Auswahl einer Variante zumindest für C-YAWL gewährleistet werden.

Die selbständige Entwicklung von Prozessvarianten ist für Fachanwender kaum möglich. Da es sich bei EPC um ein Werkzeug handelt, das auch von Personen ohne tiefgreifende technische Kenntnisse genutzt wird, könnten aufgeschlossene Anwender die notwendigen Erweiterungen eventuell selbst vornehmen; allerdings sind die dabei resultierenden Prozessdefinitionen in aller Regel nicht ausführbar. Da alle möglichen Konfigurationen vordefiniert werden, haben Fachexperten auch nicht die Option, die Konfigurationen nachträglich zu modifizieren, um eine Variante an die aktuellen Anforderungen eines Falls anzupassen. Jeder klinische Pfad als konfigurierbare Prozessdefinition definiert einen Standardablauf und eine Menge von Varianten von diesem Standard; diese Varianten sind nicht ohne weiteres auf andere Pfade übertragbar. Die Durchführbarkeit manueller Abweichungsmaßnahmen wurde in Zusammenhang mit konfigurierbaren Prozessdefinitionen bisher nicht untersucht; aufgrund der komplexen Zusammenhänge zwischen Konfigurationen und Ports liegt die Vermutung nahe, dass eine konfliktfreie Anpassung nur sehr schwierig zu bewerkstelligen ist.

3.3.4 Kontextorientierte Konfiguration von Prozessinstanzen

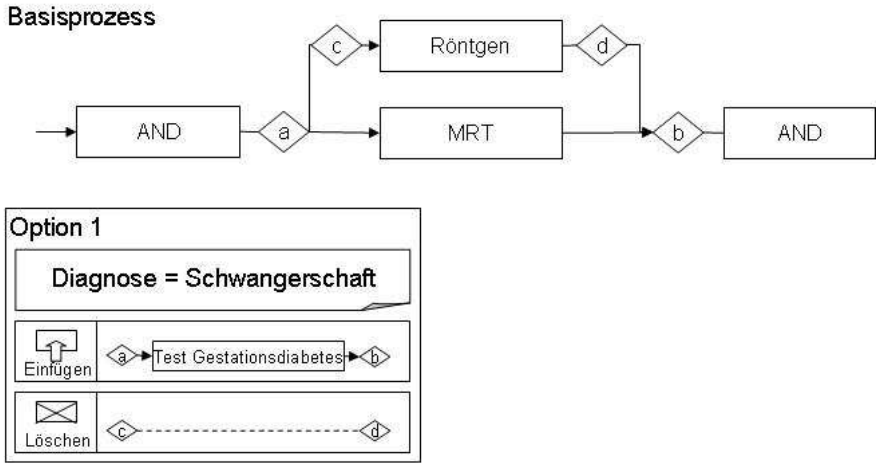
Mit Provop (Prozessvarianten mittels Optionen) präsentieren Hallerbach et al. eine Lösung, die ebenfalls gemäß dem Single-Model-Ansatz auf das Management einer großen Anzahl von Prozessvarianten in einer Prozessdefinition abzielt [Hallerbach et al. 2009a, Hallerbach 2009]. Im Gegensatz zur statischen Konfiguration zum Zeitpunkt der Instanziierung der Prozessdefinition, ermöglicht Provop jedoch die Konfiguration von Prozessen zur Laufzeit in Abhängigkeit eines sich dynamisch ändernden Kontexts. Die Anpassung erfolgt auf Basis von ad hoc Änderungsoperationen, wie z. B. den in Kapitel 3.2.6 betrachteten komplexen Änderungsoperationen. Provop unterstützt bisher die folgenden Operationen [Hallerbach et al. 2010]:

- Einfügen eines Teilprozesses
- Löschen eines Teilprozesses
- Verschieben eines Teilprozesses
- Modifikation der Attribute eines Prozesselements

Eine Menge von Änderungsoperationen kann zu einer Option zusammengefasst werden. Die Anwendung einer Menge von Optionen auf den Basisprozess resultiert wiederum in einer Prozessvariante. Auf welche Elemente der Prozessdefinition sich die Änderungsoperationen einer Option beziehen, kann entweder statisch an Hand der eindeutigen Identifizierer dieser Ele-

mente oder mit Hilfe von Variationspunkten festgelegt werden. Variationspunkte werden analog zu dem von Puhlmann et al. beschriebenen Vorgehen [Puhlmann et al. 2005] durch eine entsprechende Erweiterung des Wf-Metamodells spezifiziert. Die nachfolgende Abbildung demonstriert die Nutzung von Variationspunkten zur Anwendung von Optionen.

Abbildung 22 Modellierung von Prozessvarianten als Optionen in Provop



Der Basisprozess entspricht im Fall klinischer Pfade einer Prozessvariante, die am häufigsten verwendet wird. Diese Prozessvariante wird um mögliche Optionen erweitert. Im Beispiel sieht der Basisprozess standardmäßig für die Diagnostik eine Röntgenuntersuchung und ein MRT vor. Wenn zusätzlich die Diagnose »Schwangerschaft« gestellt wird, soll ein Test auf Gestationsdiabetes parallel zum MRT eingefügt werden, während die kontraindizierte Röntgenuntersuchung gelöscht wird. In Provop entspricht »Diagnose« einer Kontextvariablen und der Ausdruck »Diagnose = Schwangerschaft« einer Kontextbedingung, deren Erfüllung die Auswahl der entsprechenden Option zur Folge hat. Es besteht die Möglichkeit, Beziehungen zwischen Kontextvariablen zu definieren und Kontextregeln einzuführen durch die gültige von ungültigen Kombinationen von Kontextvariablen unterschieden werden können. Ändern sich die Kontextbedingungen zur Laufzeit des Prozesses, findet eine dynamische Rekonfiguration der Prozessinstanz statt [Hallerbach et al. 2008a]; d. h. Änderungen des Kontexts können die Wahl anderer Optionen als ursprünglich geplant zur Folge haben.

Gegenüber der statischen Konfiguration zum Zeitpunkt der Instanziierung, bietet Provop den Vorteil, auch zur Laufzeit auf Varianz vom klinischen Pfad zu reagieren. Welche Konfiguration gewählt wird, hängt von den während der Prozessausführung gültigen Werten der Kontextvariablen ab. Tabelle 15 zeigt, welche Anforderungen Provop an die Umsetzung von Prozessvarianten von klinischen Pfaden bereits erfüllt.

Tabelle 15 Ergebnisse der Analyse von Provop zur Unterstützung von Prozessvarianten

Zentrale Anforderung		**Resultierende Teilanforderung**		**Provop**
3.	Unterstützung von Prozessvarianten	3.1	Prozessvarianten als standardisierte Bündel von Abweichungsmaßnahmen müssen für Fachanwender modellierbar sein (siehe Kapitel 2.2.3.4.1, 2.2.3.4.4)	**Nein**

3.2	Fachanwender und WfMS müssen klar zwischen Standardprozess und Variante unterscheiden können (siehe Kapitel 2.2.3.1, 2.2.3.2)	**Ja**
3.3	Semantische Abhängigkeiten zwischen einzelnen Abweichungsmaßnahmen einer Prozessvariante müssen berücksichtigt werden (siehe Kapitel 2.2.3.4.6)	**Ja**
3.4	Fachanwender müssen Prozessvarianten in Abhängigkeit des Behandlungsfalls anpassen können (siehe Kapitel 2.2.3.4.1, 2.2.3.4.4)	**Ja**
3.5	Prozessvarianten können unabhängig von bestimmten klinischen Pfaden definiert und eingesetzt werden (siehe Kapitel 2.2.3.4.7)	**Nein**
3.6	Konflikte mit manuellen Abweichungsmaßnahmen werden vermieden (siehe Kapitel 2.2.3.4.1)	**Nein**
3.7	Die Ausführung des klinischen Pfades muss auch nach der Auswahl einer Prozessvariante möglich sein (siehe Kapitel 2.1.5)	**Ja**

Sofern der Basisprozess dem Standardvorgehen entspricht, ist die Unterscheidung zwischen Standard und Variante möglich. Durch die Zusammenfassung einzelner Änderungsoperationen zu Optionen oder die Festlegung von Beziehungen zwischen Optionen durch Spezifikation gemeinsamer Kontextparameter lassen sich semantische Abhängigkeiten zwischen den Änderungsmaßnahmen in Provop berücksichtigen. Da lediglich die Anpassung von Kontextparametern notwendig ist, um auf die konkrete Ausgestaltung einer Variante Einfluss zu nehmen, könnte diese Aufgabe von Fachexperten selbständig durchgeführt werden. Provop umfasst ein Konzept, das die korrekte Anwendbarkeit von Prozessvarianten überprüft [Hallerbach et al. 2009b]; auf Basis dieses Konzeptes lässt sich die Ausführbarkeit eines Pfades auch nach seiner Anpassung sicherstellen.

Die Definition von Optionen durch die Auswahl von Änderungsoperationen, die Einbettung von Variationspunkten sowie die Festlegung von Kontextbedingungen und –parametern sind keine trivialen Aufgaben, die effizient von Fachexperten durchgeführt werden können. Dementsprechend ist das Konzept nicht geeignet, um Anwendern die selbständige Modellierung von Prozessvarianten zu ermöglichen. Obwohl die Optionen generell unabhängig von klinischen Pfaden entwickelt werden können, müssen die Variationspunkte für jeden Pfad separat gesetzt werden; aus diesem Grund können Varianten nicht komplett unabhängig von verschiedenen Pfaden angewandt werden. Auch Konflikte mit manuellen Abweichungen lassen sich nicht vermeiden; manuelle Änderungen, die z. B. am Standardprozess durchgeführt werden, können durch die Auswahl von Prozessvarianten wieder verloren gehen.

3.3.5 Ereignisgesteuerte Umsetzung von Prozessvarianten

Grundsätzlich stellen die bisher vorgestellten Lösungsansätze hohe Anforderungen an die Planbarkeit von Varianz. D. h., nicht nur ihre Auswirkungen, sondern auch ihr Beobachtungszeitpunkt während der Patientenbehandlung müssen im Voraus bekannt sein. Wenn man nicht in der Lage ist, den Zeitpunkt, zu dem eine Varianz auftreten kann, vorherzusehen, bedeutet das im Extremfall, dass nach jeder Aktivität innerhalb der Prozessdefinition eine Abfrage stattfinden muss, um festzustellen, ob die entsprechende Varianz aufgetreten ist oder nicht. Wäh-

rend es bei Ausführungsprioritäten möglich ist, parallel zu dem gesamten Bereich in der Prozessdefinition einen Teilprozess zur Behandlung der Varianz vorzusehen, müssen bei Provop zahlreiche Variationspunkte mit der Prozessdefinition verknüpft werden. Einige Lösungsansätze setzen den Schwerpunkt auf die Tatsache, dass Varianz zwar planbar aber dennoch zu beliebigen Zeitpunkten während der Prozessausführung auftreten kann; damit werden die mit dem Begriff »Planbarkeit« verbundenen Anforderungen an entsprechende Ausnahmefälle abgemildert.

Casati et al. entwickeln einen Lösungsansatz, der den Umgang mit Varianz unabhängiger von ihrem Beobachtungszeitpunkt macht, indem sie die gesamte Abwicklung von Varianz aus der Prozessdefinition auslagern und durch die Spezifikation von ECA-Regeln (Event Condition Action Rules) abdecken [Casati et al. 1999]. Zu diesem Zweck wurde im Rahmen des WIDE Projektes eine eigene Sprache zur Abbildung planbarer Varianz, die hier vor allem unter dem Aspekt des Ausnahmefalls betrachtet wird, Chimera-Exc entwickelt. Darüber hinaus wurde ein Event Handler System aufgesetzt und beispielhaft mit einem WfMS integriert. Die mit Chimera-Exc modellierten ECA-Regeln bestehen im Wesentlichen aus drei Teilen:

- Der Ereignisteil definiert die Merkmale eines Ausnahmefalls, wie z. B. das Einfügen eines Datensatzes in die Relation eines Datenbankmanagement Systems (DBMS). Im Prinzip unterscheidet Chimera-Exc Datenmanipulationsereignisse, externe Ereignisse, temporale Ereignisse und Workflow Ereignisse.
- Bedingungen werden spezifiziert, um auszuwerten, ob es sich tatsächlich um einen Ausnahmefall handelt, der entsprechend beantwortet werden muss.
- Der Aktionsteil umfasst dann Operationen, die in Folge des Ausnahmefalls zu ergreifen sind. Chimera-Exc differenziert zwischen Aktionen zur Datenmodifikation, wie z. B. Anlegen eines neuen Datenobjekts oder Ändern der Attribute eines existierenden Objekts, und Aktionen mit Blick auf das Workflow Management, wie z. B. Benachrichtigung von Agenten, Starten neuer Subprozesse oder Abbrechen von Aktivitäten.

ECA-Regeln, die in Chimera-Exc definiert werden, haben entweder einen prozess-spezifischen oder einen globalen bzw. prozess-übergreifenden Geltungsbereich. Als Beispiele für die Notwendigkeit zur Betrachtung prozess-übergreifender Ausnahmefälle werden die Nicht-Verfügbarkeit global benötigter Ressourcen oder die Überwachung der Einhaltung global geltender Geschäftsregeln genannt [Casati et al. 2000]. Es wird erwähnt, dass der Aktionsteil einer Regel auch strukturelle Änderungen an Prozessinstanzen zur Folge haben kann; allerdings wird nicht dargelegt, wie entsprechende Aktionen spezifiziert werden und auf welche Weise die Änderungen zur Laufzeit erfolgen.

Müller et al. entwickelten einen Ansatz, der die Durchführung struktureller Änderungen an Prozessinstanzen unter Nutzung vordefinierter ECA-Regeln ermöglicht [Müller et al. 2004]. Im Unterschied zu WIDE erfolgt die Verknüpfung von Ausnahmeregeln und Prozessen nicht während der Modellierung, sondern dynamisch zur Laufzeit. Interessanterweise stützt sich die Anforderungsanalyse auf einen klinischen Behandlungsprozess zur Durchführung von Chemotherapien [Müller & Heller 1998, Müller & Rahm 1999]. Das Lösungskonzept wird für ein prototypisch entwickeltes WfMS AGENTWORK realisiert, dessen Wf-Metamodell und Ausführungssemantik weitgehend auf den Ergebnissen aus dem ADEPT2 Projekt [Reichert & Dadam 1998, Reichert 2000b] beruhen. Motivation für das Konzept sind neben der Feststellung, dass Ausnahmen zu beliebigen Zeitpunkten auftreten können, auch die Erkenntnis, dass Anwender durch geeignete Automatismen bei der Umsetzung von Anpassungen unterstützt werden müssen und dass reaktive Änderungen, die erst umgesetzt werden, sobald die Prozess-

ausführung den von der Varianz betroffenen Prozessteil erreicht, für die Flexibilisierung medizinischer Prozesse nicht ausreichen.

Für die Spezifikation der ECA-Regeln wurde eine eigene formale Logik »ActiveTFL« basierend auf der objekt-orientierten Frame-Logic [Kifer et al. 1995] und Elementen der temporalen Logik erstellt. Im Gegensatz zu den mit Chimera-Exc definierten Regeln ist es mit ActiveTFL möglich, auch zeitliche Zusammenhänge auszudrücken, z. B. wenn Ereignis *x* die nächsten sieben Tage hindurch weiterhin eintritt. ECA-Regeln in ActiveTFL haben den folgenden Aufbau [Müller et al. 2004]:

WHEN *Ereignis*
WITH *Bedingung*
THEN *Änderungsoperation*
VALID-TIME *Zeitperiode, während der das Ereignis Relevanz hat*

Änderungsoperationen entsprechen F-Logic Prädikaten und werden ähnlich zu den in ADEPT2 eingesetzten Operationen (siehe Kapitel 3.2.6). realisiert. Im Gegensatz zu Provop bezieht sich der Aktionsteil einer ECA-Regel nicht direkt auf Elemente einer Prozessdefinition, sondern stellt die Verknüpfung indirekt durch die Spezifikation so genannter Aktivitätsmuster her; mit Aktivitätsmustern können Eigenschaften von Prozesselementen spezifiziert werden, die ein konkretes Element erfüllen muss, um Gegenstand einer Modifikation zu werden. Ein mögliches Aktivitätsmuster *A* kann z. B. durch den folgenden F-Logic Ausdruck definiert werden:

A := Drug-Administration[drug = „ETOPSID“, dosage > 100, unit = mg]

Alle Aktivitäten, die der Verabreichung des Medikaments Etopsid in einer Menge, die 100 mg überschreitet, dienen, werden von dem Pattern erfasst. Auf diese Weise besteht die Möglichkeit, die Elemente auf die sich eine Änderungsoperation bezieht, dynamisch zu evaluieren. Damit bei einer Varianz, die nur temporär gültig ist, nicht alle betroffenen Prozesselemente angepasst werden, wurden Algorithmen entwickelt, die die entsprechenden Änderungsregionen unter Nutzung temporaler Zusammenhänge berechnen. Die folgende Tabelle gibt einen Überblick über die Anforderungen an die Umsetzung von Prozessvarianten, die von AGENTWORK abgedeckt werden können.

Tabelle 16 Ergebnisse der Analyse von AGENTWORK zur Unterstützung von Prozessvarianten

Zentrale Anforderung	**Resultierende Teilanforderung**		**AGENTWORK**
3. Unterstützung von Prozessvarianten	3.1	Prozessvarianten als standardisierte Bündel von Abweichungsmaßnahmen müssen für Fachanwender modellierbar sein (siehe Kapitel 2.2.3.4.1, 2.2.3.4.4)	**Nein**
	3.2	Fachanwender und WfMS müssen klar zwischen Standardprozess und Variante unterscheiden können (siehe Kapitel 2.2.3.1, 2.2.3.2)	**Ja**
	3.3	Semantische Abhängigkeiten zwischen einzelnen Abweichungsmaßnahmen einer Prozessvariante müssen berücksichtigt werden (siehe Kapitel 2.2.3.4.6)	**Ja**
	3.4	Fachanwender müssen Prozessvarianten in Abhängigkeit des Behandlungsfalls anpassen können (siehe Kapitel 2.2.3.4.1, 2.2.3.4.4)	**Nein**

3.5	Prozessvarianten können unabhängig von bestimmten klinischen Pfaden definiert und eingesetzt werden (siehe Kapitel 2.2.3.4.7)	**Ja**
3.6	Konflikte mit manuellen Abweichungsmaßnahmen werden vermieden (siehe Kapitel 2.2.3.4.1)	**Nein**
3.7	Die Ausführung des klinischen Pfades muss auch nach der Auswahl einer Prozessvariante möglich sein (siehe Kapitel 2.1.5)	**Ja**

Obwohl AGENTWORK die Durchführung komplexer Änderungen nicht explizit adressiert [Müller & Rahm 1999], kann mit dem Konzept die Umsetzung von Prozessvarianten unterstützt werden. Anwender und WfMS können zwischen Standardprozess und Variante unterscheiden, da der klinische Pfad dem nicht veränderten Basisprozess entspricht; das Auftreten von Varianz resultiert in automatisierten Änderungen am Standardablauf. Auf Basis der ECA-Regeln können Abhängigkeiten zwischen einzelnen Änderungsmaßnahmen definiert werden. Anders als bei den bisher betrachteten Lösungskonzepten erlaubt AGENTWORK außerdem die pfadunabhängige Durchführung von Anpassungen, da sich die ECA-Regeln immer auf abstrakte Aktivitätsmuster und nicht auf konkrete Aktivitäten oder sonstige Elemente in der Prozessdefinition beziehen. Da die Lösung auf der formalen Prozessmodellierungssprache von ADEPT2 und den dort implementierten Kontrollmechanismen beruht, kann die Ausführbarkeit des Prozesses auch nach erfolgter Modifikation weitgehend gewährleistet werden. Änderungen, die zu Inkonsistenzen des Kontrollflusses führen, werden z. B. automatisch verhindert.

Allerdings kann von Fachexperten nicht verlangt werden, Prozessvarianten auf Basis von ECA-Regeln selbständig zu definieren. Insgesamt ist die effiziente und kontinuierliche Pflege von einer wachsenden Menge von ECA-Regeln verbunden mit der Sicherstellung ihrer Konsistenz nicht trivial zu lösen. Dementsprechend haben die Anwender auch keine Möglichkeiten Einfluss auf die konkrete Gestaltung und Funktionsweise einer Variante zu nehmen; sie können sie lediglich annehmen oder verwerfen. Das Konzept hinter AGENTWORK ist nicht danach ausgerichtet, Konflikte zwischen manuellen Abweichungsmaßnahmen und automatisch durchgeführten Änderungen zu verhindern.

3.4 Zusammenfassung der Analyseergebnisse

Die Analyse verwandter Arbeiten im Umfeld des flexiblen Prozessmanagements hat gezeigt, dass eine ganze Reihe von Lösungsansätzen existiert, die die Durchführung von Abweichungsmaßnahmen an standardisierten Prozessdefinitionen ermöglichen. Im Prinzip lassen sich zwei grundlegende und bisher miteinander konkurrierende Vorstellungen unterscheiden: Die deklarative Prozessmodellierung zielt darauf ab, die Prozessausführung nur wenig zu reglementieren und stattdessen den Anwendern ein hohes Maß an Freiheit bei der Gestaltung der Prozessabläufe einzuräumen. Diesem Paradigma folgen Declare und FPG als wissenschaftliche Ansätze (siehe Kapitel 3.2.4) sowie auch die praktischen Lösungen (siehe Kapitel 3.1.1 und 3.1.2). Alle übrigen Verfahren gehen von einer imperativen bzw. prozeduralen Prozessmodellierung aus, die die Reihenfolge, in der Aktivitäten ausgeführt werden, weitgehend festlegt. Mit Ansätzen wie Late Binding (siehe Kapitel 3.2.2) und Late Modeling (siehe Kapitel 3.2.3) existieren auch Mischformen, bei denen die Flexibilität bisher jedoch auf vordefinierte Regionen im Prozess beschränkt ist. Das prozedurale Prozessmodellierungsparadigma erleichtert die Spezifikation des Datenflusses und die Operationalisierung von Aktivitäten, die beide zu einem hohen Anteil vom Kontrollfluss determiniert werden. Es liegt nahe, dass die

Wahrscheinlichkeit für die Notwendigkeit von Ad-hoc-Prozessänderungen bei imperativen Prozessmodellierungssprachen sehr viel höher ist, als bei deklarativen Sprachen.

Prinzipiell lässt sich feststellen, dass sich die deklarative Prozessmodellierung lohnt, wenn die allgemeine Prozessflexibilität extrem hoch ist. Auch wenn klinische Pfade initial oftmals nur ein Rahmenwerk beschreiben, in dem sich die medizinische Behandlung bewegen soll und damit eher deklarativen Ansätzen ähneln, sind die verlässliche Koordination der Aufgaben, die verbindliche Terminierung und die optimierte Planung der Ressourcenbelegung nur dann möglich, wenn der Standardablauf hier klare Vorgaben macht. Je mehr klinische Pfade als Werkzeug zur Standardisierung von Behandlungsabläufen verstanden und umgesetzt werden, desto mehr entsprechen die Prozessdefinitionen der Pfade der prozeduralen Logik. Diese Entwicklung wird durch die steigende Bedeutung der Informationstechnologie in Krankenhäusern weiter zunehmen, weil damit auch der operationale Aspekt von Prozessen an Bedeutung gewinnt. D. h. klinische Pfade werden mittelfristig neben den manuell zu erbringenden Aufgaben (»Human Tasks«) auch immer öfter der Steuerung automatisierter Aktivitäten dienen. Das bedeutet aber, dass aus deklarativen klinischen Pfaden in zunehmendem Maße prozedurale Pfade werden. Mit Systemen wie CAKE2, WASA2 (siehe Kapitel 3.2.5) und ADEPT2 (siehe Kapitel 3.2.6 und 3.3.2) stehen auch bereits Systeme zur Verfügung, mit denen sich manuelle Abweichungsmaßnahmen durch strukturelle Änderungen an prozeduralen Prozessdefinitionen und –instanzen umsetzen lassen. Durch weitergehende Ansätze wie Provop (siehe Kapitel 3.3.4) oder AGENTWORK (siehe Kapitel 3.3.5) ist sogar die automatisierte Durchführung komplexer Anpassungsfolgen als Prozessvarianten zur Laufzeit möglich.

Die kompositionale Modellierung mit der Unterscheidung zwischen Domänen- und Szenarienmodell liefert die Grundlage, wie deklarative und prozedurale Paradigmen miteinander kombiniert werden können, um den Anforderungen an eine Anwender-getriebene Entwicklung, Ausführung und Anpassung klinischer Pfade gerecht zu werden (siehe Kapitel 3.1.3). Da das Domänenmodell selbst nicht ausgeführt wird, lassen sich hier unter Nutzung deklarativer Methoden grundsätzliche Regeln definieren, die bei der Komposition von Pfadelementen zu kompletten klinischen Pfaden beachtet werden müssen. Das Szenarienmodell, also der ausführbare klinische Pfad, sollte dann jedoch einer möglichst prozeduralen Prozessdefinition entsprechen, um den Grad an Planbarkeit und Automatisierbarkeit zu maximieren. Durch die Unterstützung entsprechender Abweichungsmaßnahmen und Prozessvarianten ist aber auch bei prozeduralen Prozessen die Option zum kontrollierten Umgang mit Varianz gegeben. Es gilt also ein Lösungskonzept zu entwickeln, das das Prinzip der kompositionalen Modellierung auf die Entwicklung und dynamische Adaption klinischer Pfade überträgt, wobei die Vorteile der deklarativen und prozeduralen Paradigmen so ausgenutzt werden, dass ein schrittweiser Übergang von einer reaktiven zu einer stärker standardisierten aber ausreichend flexiblen Behandlungsplanung und –steuerung erreicht werden kann.

4 Lösungskonzept

An das Prozessmanagement im klinischen Bereich stellen sich hohe Anforderungen. Einerseits wird mit der Definition des klinischen Pfades ein Standard für die Behandlung von Patientinnen und Patienten ausgehend von einer Diagnose, einer Therapie oder einem Symptombild festgelegt. Andererseits können aufgrund der individuellen Behandlungssituation jederzeit Abweichungen vom Standard auftreten, mit denen der behandelnde Arzt umgehen muss. Das bedeutet, dass technisch wenig versierte Anwender Prozessänderungen durchführen sollen, wobei die strukturelle Korrektheit der Definition und damit die Stabilität des Gesamtsystems nicht gefährdet werden dürfen. Die Fähigkeit zur Durchführung von Änderungen an einer Prozessdefinition bzw. an ihrer Instanz setzt das Verständnis für die Struktur und Semantik der Definition voraus. Daraus folgt konsequenterweise, dass Anwender in der Lage sind, Prozesse zu modifizieren, wenn sie zuvor selbst die Prozessdefinition erstellen konnten.

Mit der Entwicklung domänenspezifischer Sprachen (DSL) wird versucht, den Vorgang der Prozessmodellierung an der Sicht der Domänenexperten und späteren Anwender auszurichten [Bentley 1986, Deursen et al. 2000]. Wie bereits in der Einleitung skizziert, zielt das gemeinschaftlich von drei Fraunhofer Instituten bearbeitete SPOT-Projekt auf die Schaffung einer Modellierungsumgebung ab, die sich an den Bedürfnissen der Anwender aus der Gesundheitsdomäne orientiert und das ärztliche Personal in die Lage versetzt, Prozesse für die integrierte Versorgung aus ihrer Sicht zu modellieren, auszuführen und zu verändern (siehe Kapitel 1.2). Im Rahmen des SPOT-Projektes wurde bereits eine prototypische Realisierung für die Modellierung und dynamische Adaption von Behandlungsprozessen kreiert. Dieser Prototyp zeigt bisher jedoch nur beispielhaft, wie die Vision hinter SPOT, nämlich die Modellierung ausführbarer Prozesse durch Fachanwender, in die Realität umgesetzt werden kann. Voraussetzung für dieses Vorgehen ist die Schaffung einer Bibliothek über das prozessorientierte Domänenwissen, die die Anwender nutzen können, um aus ihrer Sicht ausführbare Prozesse zu erstellen. Diese Arbeit soll die theoretischen und formal fundierten Grundlagen für eine solche Wissensbasis schaffen. Darüber hinaus soll spezifiziert werden, nach welchen Regeln Prozesse unter Nutzung des Domänenwissens abgeleitet und auch nach der Instanziierung dynamisch modifiziert werden können.

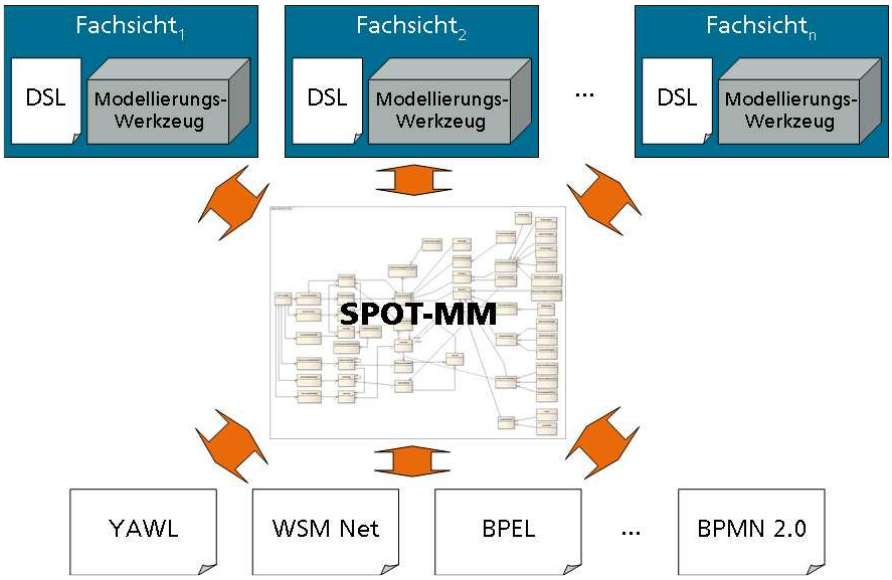
In diesem Kapitel wird zunächst ausgehend von den Ergebnissen des SPOT-Projektes der Bedarf für die Entwicklung eines generischen Konzeptes zur Modellierung und dynamischen Adaption klinischer Pfade auf Basis semantischer Prozessfragmente identifiziert. Anschließend wird eine Übersicht über die verschiedenen Phasen des Lösungskonzeptes und die notwendigen Bausteine gegeben. Ausgehend von den einzelnen Phasen werden die Lösungsbausteine anschließend detailliert beschrieben.

4.1 SPOT-Technologie: Lösung für ein anwendernahes Prozessmangement

Hintergrund des SPOT-Projektes ist die Vision, Domänenexperten die geeigneten Mittel an die Hand zu geben, Geschäftsprozesse aus ihrer Sicht zu beschreiben und möglichst automatisch zur Ausführung zu bringen. Da Geschäftsprozesse in hoch dynamischen Domänen wie dem Gesundheitswesen keine starren Abläufe repräsentieren, sondern vielmehr flexibel an die aktuell geltenden Rahmenbedingungen angepasst werden müssen, sollen auch die Fachexperten die Möglichkeit haben, Prozessmodelle oder bereits laufende Prozesse selbständig zu

modifizieren. Weil von den Anwendern nicht erwartet werden kann, technisch geprägte Modellierungsumgebungen zu nutzen, um Prozessdefinitionen zu kreieren, die von WfMS interpretiert und verarbeitet werden können, sind domänenspezifische Sprachen und Werkzeuge zu entwickeln, die sich speziell an den Anforderungen der jeweiligen Domäne und ihrer Anwender orientieren. Diese Kombination aus DSL und Modellierungswerkzeug wird im SPOT-Projekt auch als »Fachsicht« bezeichnet. Um Prozesse, die in einer Fachsicht spezifiziert worden sind, zur Ausführung bringen zu können, ist die Transformation in eine Sprache erforderlich, die von einem WfMS zur Steuerung des spezifizierten Prozesses genutzt werden kann. Bei solchen Sprachen handelt es sich z. B. um YAWL [van der Aalst & ter Hofstede 2002, ter Hofstede et al. 2010], die im ADEPT2 Projekt spezifizierte Sprache für WSM Nets [Rinderle et al. 2004] oder auch WSBPEL [Andrews et al. 2003, OASIS 2007] und BPMN 2.0 [OMG 2009]. Damit nicht für jede Fachsicht Transformationsregeln in jede beliebige ausführbare Sprachen erstellt werden müssen, gibt es in SPOT ein übergeordnetes Metamodell »SPOT-MM« (siehe Abbildung 23).

Abbildung 23 SPOT-MM als Basis für Transformationen zwischen DSL und ausführbaren Prozessmodellierungssprachen



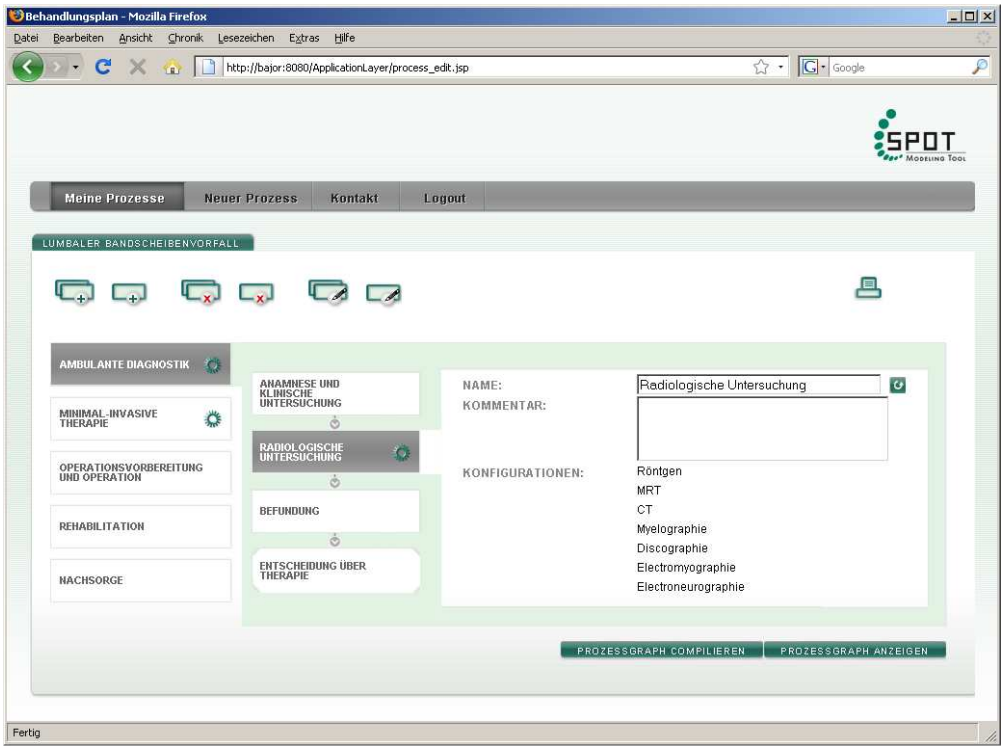
Solange sich die DSL einer Fachsicht auf SPOT-MM abbilden lässt, ist also die Transformation in alle ausführbaren Sprachen, die von SPOT unterstützt werden möglich. Auf diese Weise lassen sich unterschiedliche WfMS nutzen, um fachlogische Prozesse zu steuern. Diese Entkopplung ermöglicht Unternehmen nicht nur eine größere Unabhängigkeit gegenüber oftmals externen IT-Experten, da sie nun nicht mehr gezwungen sind, alle sensiblen Details der Geschäftsprozesse ihrer Einrichtung offen zu legen; sie erhöht auch die Freiheit in der Wahl der Hersteller prozessorientierter Systeme, da an den fachlogischen Ablaufbeschreibungen nichts geändert werden muss, um sie von einem anderen WfMS ausführen zu lassen.

4.1.1 DSL für integrierte Versorgungsprozesse

Im Rahmen des SPOT-Projektes wurde ein Prototyp, der »SPOT Healthcare Demonstrator« entwickelt, um zu zeigen, dass diese Vision tatsächlich Realität werden kann [Reuter 2009].

Als Anwendungsfall dient der in Kapitel 2.2.1.1 skizzierte Behandlungsprozess für Patientinnen und Patienten mit lumbalem Bandscheibenvorfall. Für die Fachsicht wurde eine DSL für so genannte Behandlungspläne erstellt [Neuhaus et al. 2010]. Behandlungspläne beschreiben an Hand einer Kette von Behandlungsschritten den kompletten Versorgungsablauf. Da Behandlungspläne nicht nur für das medizinische Personal, sondern vor allem für Patienten verständlich sein sollen, werden im Demonstrator stets nur solche Behandlungsschritte dargestellt, die für den Patienten Relevanz haben. Daher handelt es sich bei Behandlungsplänen um eine Konstruktion, die von der Anwendung her dem Patientenpfad nach Hellmann entspricht: »... heruntergebrochene klinische Pfade, die in übersichtlicher und für den Patienten verständlicher Form verdeutlichen, was warum in welchem Behandlungsschritt mit ihm passiert« [Hellmann & Eble 2009]. Die einzelnen Behandlungsschritte repräsentieren singuläre Aktivitäten innerhalb des Geschäftsprozesses, wie z. B. »Anamnese« oder »Radiologische Untersuchung«. Da es sich beim Patienten um das kritische Element handelt, das in jedem Behandlungsschritt zugegen sein muss, können sämtliche Behandlungsschritte in sequentieller Form über Kontrollflüsse miteinander verbunden werden. Die Tatsache, dass es innerhalb der Behandlungspläne keine Parallelismen gibt, erleichtert die visuelle Darstellung enorm. Abbildung 24 zeigt dies am Beispiel eines Screenshots des »SPOT-Care-Plan-Modelers«, eines webbasierten Modellierungswerkzeugs für Behandlungspläne.

Abbildung 24 Modellierung von Behandlungsplänen mit Hilfe des SPOT-Care-Plan-Modelers

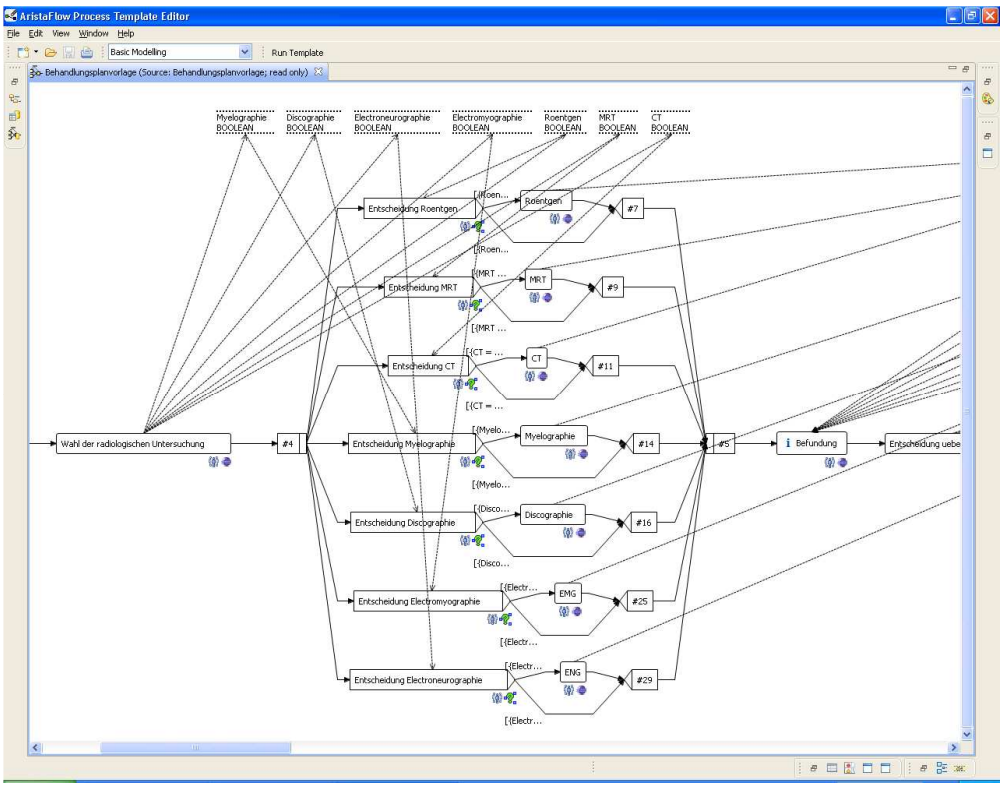


Eine Kette von Behandlungsschritten kann wiederum zu einem Behandlungsabschnitt zusammengefasst werden. Diese Notation kommt der Unterteilung medizinischer Versorgungsprozesse in Phasen wie »Ambulante Diagnostik«, »Operationsvorbereitung und Operation«, usw. entgegen. Jeder Behandlungsabschnitt endet mit einem Meilenstein (achteckiges Symbol in der Abbildung). Dabei handelt es sich um eine besondere Form von Behandlungsschritt, da im

Meilenstein entschieden wird, welcher Behandlungsabschnitt als nächstes gewählt werden soll. Stehen mehrere Abschnitte als Alternativen zur Verfügung entspricht der Meilenstein in der Terminologie von WfMS einer bedingten Verzweigung (»Exclusive Choice« Konstrukt, siehe Kapitel 3.3.1). Das Konzept der Behandlungspläne trägt auch dem Umstand Rechnung, dass es in der Medizin eine Reihe gleichartiger Leistungen gibt, die in Abhängigkeit des jeweiligen Behandlungsfalles sehr unterschiedliche Ausprägungen annehmen können. Beispiele im Hinblick auf die stationäre Behandlung von Wirbelsäulenerkrankungen sind die radiologische Untersuchung und die Form der Therapie. Je nach konkreter Diagnose, Gesundheitszustand und persönlichen Wünschen des Patienten können bei den radiologischen Untersuchungen unterschiedliche Bild gebende Verfahren zur Anwendung kommen. Standardmäßig werden zur Abklärung eines Verdachts auf Bandscheibenvorfall eine Röntgenuntersuchung und ein MRT durchgeführt. Bei bestimmten Patienten sind jedoch andere oder ergänzende Untersuchungen, wie z.B. ein CT, notwendig. Da die unterschiedlichen Leistungsangebote bereits im Voraus bekannt sind, können sie als Konfigurationsmöglichkeiten eines Behandlungsschrittes im Behandlungsplan-Konzept berücksichtigt werden. In der Fachsicht für Behandlungspläne wurde eine Checklisten-artige Darstellung für Konfigurationsmöglichkeiten gewählt, wie Abbildung 24 am Beispiel der radiologischen Untersuchung zeigt. Konfigurationsmöglichkeiten können in ausführbaren Prozessmodellierungssprachen als »Multi-Choice« Pattern oder durch Kombination der Pattern »Parallel Split« und »Exclusive Choice« realisiert werden. Prinzipiell besteht jedoch auch die Möglichkeit, z. B. Late Composition als Sonderform des Late Modeling Ansatzes zu verwenden (siehe Kapitel 3.2.3). Voraussetzung für die Wahl des Verfahrens ist natürlich, dass das genutzte WfMS das jeweilige Konzept interpretieren kann. Abbildung 25 illustriert die Umsetzung des Behandlungsschrittes »Radiologische Untersuchung« im Modellierungswerkzeug der Aristaflow® BPM Suite[33], dem kommerziellen Produkt auf Basis des ADEPT2 Prototypen.

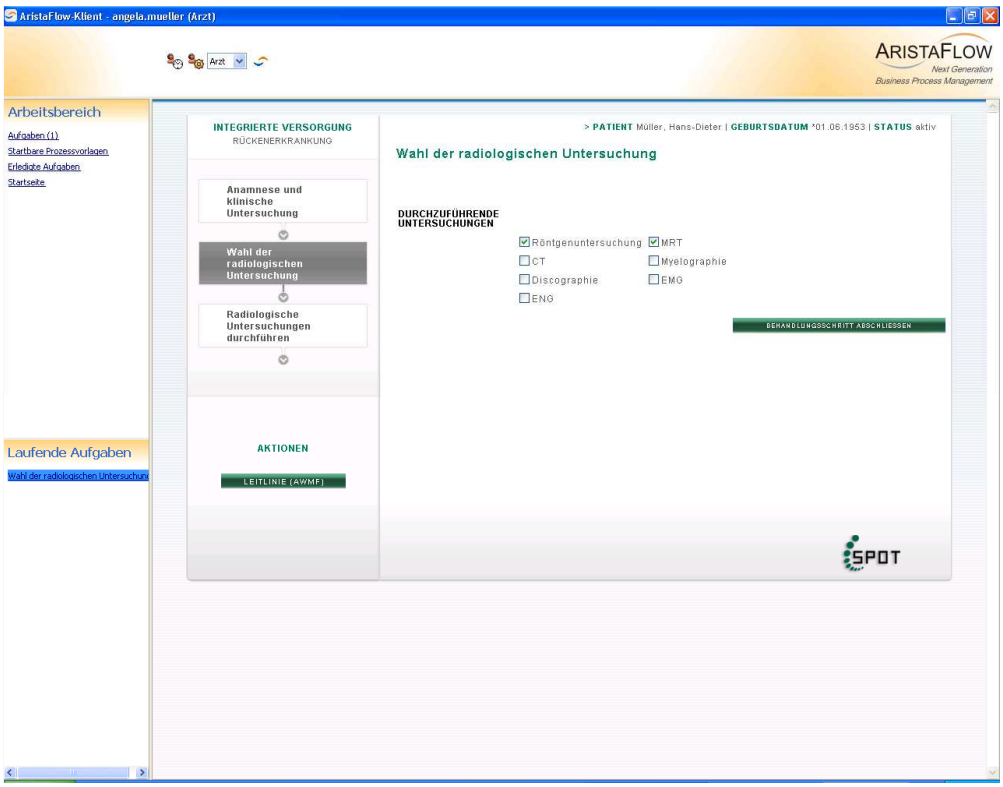
[33] Siehe Produkt Homepage unter http://www.aristaflow.com/

Abbildung 25 Repräsentation von Behandlungsplänen als ausführbare WSM Nets im Aristaflow® Template Editor



Bei einer Gegenüberstellung von Abbildung 24 und Abbildung 25 wird deutlich, in welcher Weise die Fachsicht von der formalen Prozessdefinition abstrahiert. Neben der strukturellen Gestaltung, werden in der Fachsicht auch Elemente und Zusammenhänge, die für die Anwender entweder ohnehin selbstverständlich oder irrelevant sind, nicht angezeigt. Dabei handelt es sich z. B. um Datenflüsse, Entscheidungsparameter, organisatorische Zuordnungen und Anwendungskomponenten. Mit Hilfe von SPOT-MM erfolgt die Transformation des in der Fachsicht modellierten Behandlungsplanes in eine WSM-Net konforme Notation, die anschließend durch den Aristaflow® Server ausgeführt und im Client dargestellt werden kann (siehe Abbildung 26).

Abbildung 26 Ausführung von Behandlungsplänen mit Hilfe der Aristaflow® BPM Suite



Für diese Umwandlung ist im SPOT Healthcare Demonstrator eine spezielle Komponente, der »SPOT-Process-Compiler« zuständig. Dieser nimmt aktuell eine Prozessdefinition in SPOT-MM Notation entgegen und erzeugt daraus in Abhängigkeit des gewählten WfMS die ausführbare Prozessdefinition.

4.1.2 Kompositionale Prozessmodellierung

Ein grundlegendes Konzept, das als zentrale Klasse im Klassendiagramm von SPOT-MM vorgesehen ist, ist das »Prozessfragment«. Bei einem Prozessfragment handelt es sich um eine Komponente, deren Schnittstelle und deren Implementierung bereits klar definiert sind. Die Schnittstellenspezifikation umfasst zwingend die Datenobjekte, die von dem Fragment konsumiert bzw. produziert werden. Optional können weitere Aspekte modelliert werden, wie z. B. erforderliche Ressourcen (Personen, Geräte, Räume) oder zeitliche Beschränkungen. Die Implementierung legt fest, auf welche Art und Weise ein Prozessfragment operationalisiert ist; die Implementierung kann demnach z. B. dem Aufruf einer Methode einer externen Anwendungskomponente entsprechen. Nicht nur die Schnittstellenspezifikation sondern auch die Implementierung sollte losgelöst von einer konkreten Plattform und einem konkreten WfMS erfolgen, um die Systemunabhängigkeit zu gewährleisten; dies kann geschehen, indem die Implementierung z. B. auf Basis der Web Service Technologie erfolgt. Prinzipiell ist es auch möglich, ein Prozessfragment als Subprozess zu spezifizieren; wird bei der Modellierung des Subprozesses jedoch eine spezielle Modellierungssprache genutzt, so kann das Prozessfragment anschließend nur von solchen WfMS ausgeführt werden, die die gewählte Sprache verarbeiten können. Um dies zu verhindern, sollten Subprozesse also auf Basis von SPOT-

MM erstellt werden. Dabei entspricht jede Aktivität des Subprozesses wiederum einem Prozessfragment. Die Umwandlung der Prozessdefinition von SPOT-MM in eine WfMS-spezifische Definition muss in Abhängigkeit des WfMS und der dort verwendeten Modellierungssprache erfolgen.

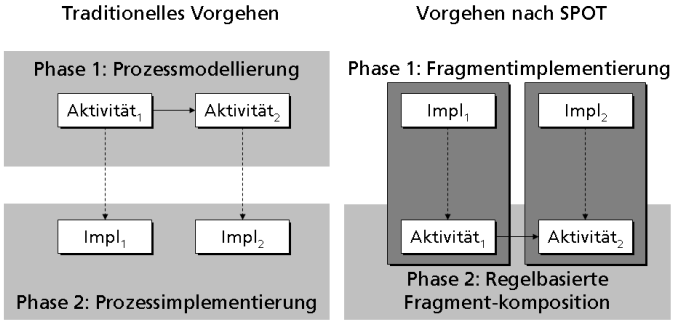
Mit der Einführung des Konzeptes der Prozessfragmente, verhalten sich alle Anwendungskomponenten, die für die Durchführung der gewünschten Aktionen zuständig sind, gegenüber dem WfMS wie Prozeduraufrufe in üblichen Programmiersprachen; d. h. der Implementierungsteil eines Prozessfragments wird mit den gemäß Schnittstellenspezifikation benötigten Datenobjekten als Parameter aufgerufen und erhält die ebenfalls in der Spezifikation festgelegten Datenobjekte als Rückgabewert. Die konkrete Implementierung des Prozessfragments bleibt auf Ebene des Geschäftsprozesses, der das Prozessfragment umfasst, verborgen.

Nach dieser Beschreibung weist das in SPOT vorgestellte Konzept der Prozessfragmente Parallelen zu den in ADEPT2 definierten »Aktivitätsvorlagen« [Reichert 2000b] auf. Aktivitätsvorlagen erhalten ebenfalls eine Schnittstellenspezifikation und kapseln die Implementierung durch eine Anwendungskomponente oder interne Systemfunktionen von ADEPT2. In ADEPT2 werden diese Vorlagen in einem Repository verwaltet und können zur Operationalisierung von Prozessaktivitäten genutzt werden. Auf diese Weise wird die Wiederverwendbarkeit der bereits entwickelten ausführbaren Komponenten ermöglicht. Grundsätzlich wird mit den in ADEPT2 definierten Aktivitätsvorlagen kein Anspruch auf die Nutzung im Kontext unterschiedlicher WfMS erhoben. Daher werden Aktivitätsvorlagen auch zur Auswertung einer prädikativen Bedingung durch ADEPT2 oder den Aufruf eines ADEPT2 Subprozesses genutzt, während es in SPOT uneingeschränkt notwendig ist, Prozessfragmente tatsächlich systemunabhängig zu implementieren.

ADEPT2 unterstützt wie die meisten WfMS die automatische Operationalisierung von Prozessaktivitäten, indem bereits vorhandene Aktivitätsvorlagen aus einem Repository ausgewählt und an der gewünschten Stelle in die Prozessdefinition eingebettet werden. Dieser Vorgang erhebt an die Ersteller einer neuen Prozessdefinition jedoch die Anforderung, dass strukturelle Abhängigkeiten und Datenabhängigkeiten zu anderen, bereits operationalisierten Aktivitäten manuell berücksichtigt werden; dies wiederum erfordert das tiefgreifende Verständnis um die konkrete Operationalisierung einer Aktivitätsvorlage und ihre Schnittstellenspezifikation. Genau dieser Umstand kann jedoch nicht als gegeben vorausgesetzt werden, wenn Fachanwender Prozessdefinitionen in Eigenregie kreieren bzw. modifizieren sollen.

In SPOT erfolgt die Prozessmodellierung auf Basis der bereits operationalisierten Prozessfragmente. Der wesentliche Unterschied zwischen SPOT und dem in ADEPT2 wie auch von anderen WfMS verfolgten Vorgehen besteht nun in der Tatsache, dass der Kontext, in dem ein Prozessfragment in eine Prozessdefinition integriert werden kann, bereits vordefiniert ist. Die Erstellung von Prozessdefinitionen entspricht daher nicht dem traditionellen Ansatz der Prozessmodellierung, sondern vielmehr der Komposition von Prozessfragmenten gemäß fest vorgegebenen Regeln. Die Abgrenzung des in SPOT verfolgten Ansatzes dieser kompositionalen Prozessmodellierung vom traditionellen Vorgehen ist in der nachfolgenden Abbildung dargestellt.

Abbildung 27 Abgrenzung der kompositionalen Prozessmodellierung in SPOT vom traditionellen Vorgehen



Das traditionelle Vorgehen bei der Prozessmodellierung und –implementierung sieht vor, dass zunächst der Geschäftsprozess entworfen wird. Dieser besteht aus Aktivitäten, die über Kontrollflüsse miteinander verbunden sind und zwischen denen Datenabhängigkeiten definiert werden. Die neuen Aktivitäten werden direkt während der Prozessmodellierung oder nachträglich im Zuge der Prozessimplementierung mit den für die Ausführung notwendigen Anwendungskomponenten verknüpft. Hierbei werden entweder vorhandene Implementierungen genutzt, wie z. B. Aktivitätsvorlagen in ADEPT2, oder neue Implementierungen entwickelt. Bei der Operationalisierung von Prozessaktivitäten müssen dem Modellierer Abhängigkeiten und Wechselwirkungen zwischen den Anwendungskomponenten bekannt sein; nur so kann sichergestellt werden, dass der modellierte Prozess im Anschluss auch wirklich ausführbar ist.

Da dieses Wissen für Fachanwender ohne tief greifende IT-Kenntnisse ein nicht zu überwindendes Hindernis darstellt, wurde bei SPOT eine alternative Vorgehensweise gewählt. Zuerst führen die IT-Experten die Implementierung durch, indem sie die notwendigen Prozessfragmente entwickeln. Zusätzlich definieren sie aber auch generelle Regeln, nach denen die Prozessfragmente miteinander kombiniert werden können. Diese Regeln können z.B. auf der Feststellung von Datenabhängigkeiten beruhen oder – in Absprache mit dem Fachpersonal – auch semantischen Ursprungs sein; z.B. wenn eine radiologische Untersuchung vorher die Gabe eines Kontrastmittels erfordert. Auf Basis der vorhandenen Prozessfragmente und der Regelmenge sind die Anwender anschließend in der Lage, die von ihnen gewünschten Prozessdefinitionen zu erstellen. Die Komposition der Prozessfragmente gemäß den vordefinierten Regeln sorgt dafür, dass immer ein ausführbarer Prozess entsteht.

Dieser Ansatz der kompositionalen Prozessmodellierung setzt aber voraus, dass in einer Einrichtung bereits Prozesswissen vorhanden ist. Im Idealfall hat ein Krankenhaus schon Erfahrungen mit der Erhebung von Prozessabläufen und der Erstellung von Prozessdefinitionen gemacht. Auf Basis dieser Prozessdefinitionen und der Leistungsbeschreibungen einzelner Abteilungen ist es anschließend möglich, wiederkehrende Muster zu identifizieren und dafür Prozessfragmente zu erstellen. Wie bereits erwähnt, kann es sich bei einem Prozessfragment sowohl um eine einzelne Funktion eines externen Anwendungssystems handeln, als auch um einen kompletten Subprozess. Wichtig ist die Feststellung, dass die Schnittstellenspezifikation der Prozessfragmente statisch ist und weder bei der Modellierung noch bei der dynamischen Adaption von Prozessen verändert wird. Prozessfragmente können dann mit der Metapher einer »Black Box« umschrieben werden, deren Struktur einmal festgelegt wird und die den Fachanwendern anschließend als Modellierungsbausteine zur Verfügung gestellt werden. Die technische Umsetzung von Prozessfragmenten ist Aufgabe von IT-Experten. Ein wesentlicher Vorteil der kompositionalen Prozessmodellierung gegenüber dem bisherigen Ablauf besteht darin, dass sich der Abstimmungs- und Kommunikationsaufwand zwischen Domänen- und IT-

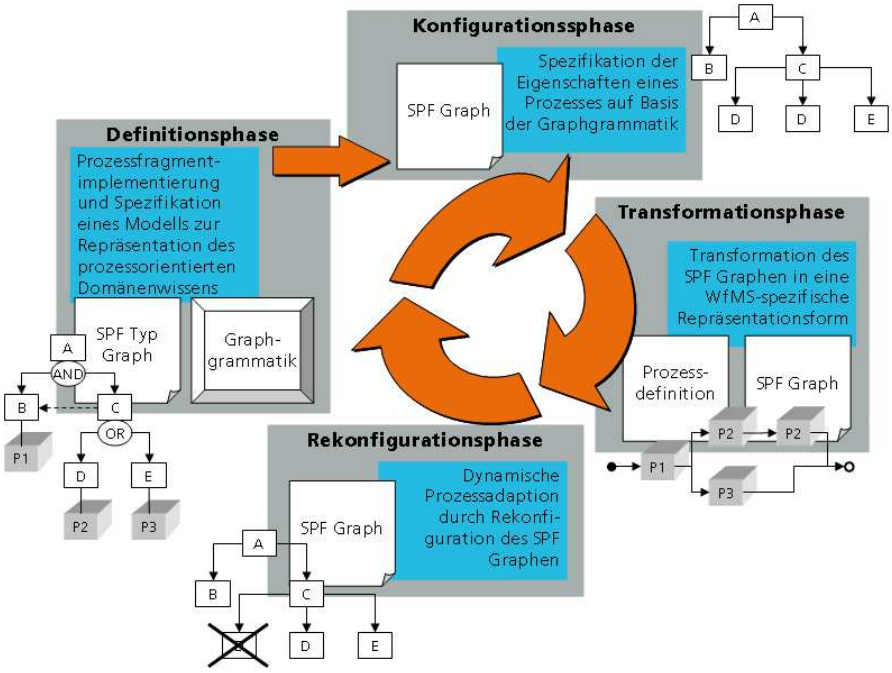
Experten nun auf klar abgrenzbare Prozessfragmente und Kompositionsregeln, die generell Gültigkeit besitzen, reduziert. Der komplexe Prozess, der sich über viele Abteilungen und Professionen erstrecken kann und daher eine Vielzahl von Schnittstellen besitzt, muss von IT-Experten nicht mehr komplett durchdrungen werden.

Auf Grund seiner visionären Ausrichtung wurde im SPOT-Projekt auf eine konkrete Spezifikation von Prozessfragmenten verzichtet. Darüber hinaus wurde kein formales Modell für Kompositionsregeln entwickelt, das den Domänenexperten die Prozessmodellierung auf Basis der Fragmente erleichtert. Werden die Prozessfragmente lediglich in einem Repository verwaltet und den Fachanwendern angeboten, ohne dass deutlich wird, in welchem hierarchischen oder strukturellem Zusammenhang die Fragmente stehen, stellt die kompositionale Prozessmodellierung keine große Arbeitserleichterung für die Domänenexperten dar. Vielmehr bedarf es klarer Regeln, die ihr Fundament aus dem prozessorientierten Domänenwissen beziehen und die definieren, welche Kompositionen von Prozessfragmenten korrekt sind bzw. die die Anwender von der müßigen Komposition der Fragmente weitgehend entlasten. Ziel dieser Arbeit ist es also ein formal fundiertes Modell für die Verwaltung des prozessorientierten Domänenwissens sowie Regeln zur Ableitung gültiger Prozessdefinitionen zu spezifizieren. Was im SPOT-Projekt bisher ebenfalls fehlt, ist die Schaffung eines generischen Konzepts zur dynamischen Änderung von Prozessdefinitionen und ihren Instanzen auf Basis von Prozessfragmenten.

4.2 Überblick über das Lösungskonzept

Das in dieser Arbeit entwickelte Lösungskonzept zur fragmentorientierten Prozessmodellierung und -adaption basiert auf der Anreicherung des im SPOT-Projekt eingeführten Konzeptes »Prozessfragment« um eine semantische Komponente. Diese semantische Komponente ergibt sich durch die Einbettung des Prozessfragments in seinen hierarchischen und strukturellen Kontext mit anderen Prozessfragmenten. Auf diese Weise wird aus einem Prozessfragment, das bisher lediglich aus Schnittstellenspezifikation und Implementierung besteht, ein so genanntes »semantisches Prozessfragment« oder kurz »SPF«. In welchen Zusammenhängen Prozessfragmente stehen, wird durch die explizite Modellierung des prozessorientierten Domänenwissens ausgedrückt. Wie in Kapitel 4.3 ausführlich beschrieben wird, lässt sich dieses Domänenwissen in Form eines Graphen abbilden. Dieser Graph legt fest, welche Prozessfragmente für die Prozessmodellierung und dynamische Adaption zur Verfügung stehen, indem er die Hierarchie und bei Bedarf auch die semantische Vernetzung der Prozessfragmente beschreibt. Da es sich hierbei um einen speziellen Graphen handelt, dessen Aufbau sich an formal definierten Regeln orientiert, wird für den Graphen der Begriff »SPF-Typ-Graph« eingeführt. SPF-Typ-Graphen entsprechen in ihrer Struktur wurzel-basierten, gerichteten und azyklischen Graphen. Die Spezifikation des SPF-Typ-Graphen durch die Erhebung und Modellierung des Prozesswissens innerhalb einer Domäne erfolgt im Rahmen der »Definitionsphase« (siehe Überblick in Abbildung 28). Ergebnis dieser Phase ist ein SPF-Typ-Graph, der bestimmten formalen Korrektheitskriterien unterliegt, und aus dem eine spezifische Graphgrammatik gewonnen werden kann.

Abbildung 28 Überblick über das Lösungskonzept



Die Graphgrammatik lässt sich im Stil der generischen Programmierung automatisch auf Basis des SPF-Typ-Graphen erzeugen. Sie ist grundlegend für die Ableitung konkreter Prozessdefinitionen aus dem allgemeinen Prozesswissen, das durch den SPF-Typ-Graphen repräsentiert wird. Der erste Schritt zur Erzeugung einer Prozessdefinition besteht in der Entwicklung eines »SPF-Graphen« durch Anwendung der Regeln der Graphgrammatik. Bei SPF-Graphen handelt es sich um typisierte Graphen, die ausschließlich nach dem Muster der SPF-Typ-Graphen gebildet werden dürfen. Auf Grund der Verfügbarkeit einer Graphgrammatik kann dies auch sicher gewährleistet werden. Die Auswahl und Anwendung der Kompositionsregeln der Graphgrammatik erfolgt in einer separaten Phase, der »Konfigurationsphase« (siehe Kapitel 4.4). Ergebnis dieser Phase ist ein Modell, das ähnlich wie SPOT-MM unabhängig von einer konkreten und gegebenenfalls WfMS-spezifischen Modellierungssprache ist und das die Eigenschaften und grundlegende Struktur einer Prozessdefinition repräsentiert. Ein wesentlicher Unterschied zu SPOT-MM und den Ansätzen in Zusammenhang mit Prozesskonfiguration (siehe Kapitel 3.3.3 und Kapitel 3.3.4) besteht darin, dass es sich nicht um eine Prozessmodellierungssprache im traditionellen Sinne handelt; das bedeutet, SPF-Graphen erheben nicht wie übliche Modellierungssprachen den Anspruch, alle prozessrelevanten Informationen in einem einzigen Modell zusammenzuführen. Stattdessen lassen sich SPF-Graphen auf semantisch erweiterte Angaben zum funktionalen Aspekt eines Workflows beschränken; optional können durch die Definition von Constraints im SPF-Typ-Graphen aber auch semantische Abhängigkeiten definiert werden, die die Anordnung der Kontrollflüsse in der resultierenden Prozessdefinition beeinflussen. Neben dieser generellen Unterscheidung verfügen SPF-Graphen, anders als SPOT-MM konforme Modelle, durch die Existenz von Graphgrammatiken außerdem um eine formal definierte Grundlage. Nichtsdestotrotz behält SPOT-MM auch nach der Einführung von SPF-Graphen seine Berechtigung, z. B. für die Spezifikation von Prozessfragmenten als WfMS-unabhängige Subprozesse.

Während SPF-Typ-Graphen also einen Möglichkeitsraum über sämtliche Prozessdefinitionen beschreiben, schränken SPF-Graphen diesen Möglichkeitsraum komplett oder in Teilen ein,

um eine spezifische Prozessdefinition zu erhalten. In der »Transformationsphase« erfolgt dementsprechend die Umwandlung des, von konkreten Modellierungssprachen, abstrahierenden SPF-Graphen in eine Prozessdefinition, die den Vorgaben einer speziellen Modellierungssprache unterliegt (siehe Kapitel 4.5). Dabei können zwei Strategien unterschieden werden, deren Wahl unter anderem von der Ausdrucksmächtigkeit der Modellierungssprache abhängt. So kann die Transformation in mehreren Schritten jeweils nach der Anwendung einer Produktionsregel durchgeführt werden. Dieses Vorgehen hat den Vorteil, dass der Anwender sofort den resultierenden Prozess sieht. Ein Nachteil besteht darin, dass der Möglichkeitsraum nach der Anwendung der ersten Regeln immer noch sehr groß ist; aus diesem Grund können auch die Ausmaße der Komplexität der entstehenden Prozessdefinition unangemessen hoch sein. Verfügt die Prozessmodellierungssprache nicht über die Mittel diese Komplexität zu verbergen, z. B. durch Nutzung des »Worklet« Konzeptes (siehe Kapitel 3.2.2) oder der »Pockets of Flexibility« (siehe Kapitel 3.2.3), die die konkrete Ausprägung eines übergeordneten, bislang unkonfigurierten Knotens im SPF-Graphen geeignet verbergen könnten, sollte dieses Verfahren nicht angewendet werden. Stattdessen sollte die Transformation erst dann erfolgen, sobald der Möglichkeitsraum des SPF-Graphen im Zuge der Konfiguration auf einen adäquaten Umfang reduziert wurde. Dieser adäquate Umfang kann z. B. durch Festlegung der Hierarchiestufe im Graphen bis zu der mindestens konfiguriert werden muss, definiert werden. Im Prinzip entspricht der Transformationsvorgang dem Paradigma des »Model-driven architecture« (MDA) oder spezieller, der generativen Programmierung. Dieser Ansatz ermöglicht die automatische Generierung von Programmcode auf der Grundlage eines formalen Modells [Czarnecki & Eisenecker 2000]; analog dazu dient das Lösungskonzept der automatischen Erzeugung von Prozessdefinition auf Basis von SPF-Graphen.

Das Lösungskonzept stellt grundsätzlich nicht die Forderung auf, dass es sich bei der Prozessmodellierungssprache, in die transformiert werden soll, um eine Sprache handelt, mit der ausschließlich Prozessdefinitionen erzeugt werden, die ausführbar und deren Korrektheit formal nachweisbar ist, wie dies z. B. bei YAWL oder ADEPT2 WSM Nets der Fall ist. Ruft man sich jedoch die Vision hinter SPOT ins Gedächtnis, der zur Folge Fachanwender in die Lage versetzt werden sollen, ausführbare Prozessdefinitionen zu kreieren, ist die Wahl solcher Modellierungssprachen deutlich von Vorteil. Obwohl also die Transformation in Abhängigkeit der gewählten Modellierungssprache erfolgen muss, wird im Rahmen des Lösungskonzeptes am Beispiel der ADEPT2 WSM Nets gezeigt, wie diese Umwandlung konkret durchgeführt werden kann. Gegenüber anderen Modellierungssprachen erleichtert ADEPT2 durch die Bereitstellung der bereits in Kapitel 3.2.6 angesprochenen komplexen Änderungsoperationen den Transformationsvorgang sehr. Obwohl in dieser Arbeit die direkte Transformation in eine ausführbare Modellierungssprache beschrieben wird, kann es auch sinnvoll sein, eine Umwandlung von SPF-Graphen in SPOT-MM konforme Modelle vorzunehmen. Zum einen besteht anschließend die Möglichkeit, Prozessdefinitionen in allen Sprachen zu erzeugen, die von SPOT unterstützt werden, zum anderen können auch domänenorientierte Fachsichten generiert werden, die auf geeignete Weise von den technischen Details einer Prozessdefinition abstrahieren. Da dieses Vorgehen durch den zusätzlichen Transformationsschritt die Komplexität und damit die Fehleranfälligkeit bei der Transformation generell jedoch erhöhen würde und Fachsichten bei Bedarf auch direkt auf SPF-Graphen definiert werden könnten, wird diese Möglichkeit im Rahmen des Lösungskonzeptes nicht weiter vertieft. Schlussendlich steht mit ADEPT2 außerdem ein adaptives WfMS zur Verfügung, so dass auch die dynamische Adaption von Prozessdefinitionen bzw. –instanzen beispielhaft für eine konkrete Prozessmodellierungssprache durchgespielt werden kann.

Das Lösungskonzept sieht für die Anpassung von Prozessen nach der Transformation des SPF-Graphen eine eigene Phase, die »Rekonfigurationsphase« vor (siehe Kapitel 4.6). Wie die Bezeichnung und das zyklische Ineinandergreifen der Phasen in Abbildung 28 bereits nahe legen, findet die Änderung also nicht direkt an der Prozessdefinition statt, sondern vielmehr am SPF-Graphen, der zu diesem Zweck rekonfiguriert werden muss. Die Rekonfiguration kann dabei unterschiedliche Ausprägungen annehmen. Eine Möglichkeit besteht im kompletten Ersetzen eines Knotens innerhalb des SPF-Graphen einschließlich des mit dem Knoten assoziierten Teilgraphen. In einer zweiten Variante ist die Anpassung fein granularer, z. B. werden nur einzelne Nachfolger neu eingefügt oder gelöscht; die restliche Struktur des von der Änderung betroffenen Teilgraphen bleibt hingegen unberührt. Für die Identifikation des von der Anpassung betroffenen Teilgraphen kommt als formale Methodik der Graphmorphismus zum Einsatz. Ein Graphmorphismus beschreibt das identische Abbild des gesuchten, zu ändernden Graphen innerhalb des SPF-Graphen. Existieren mehrere Graphmorphismen erstreckt sich die Änderung entweder auf alle betroffenen Teilgraphen oder es wird eine Einschränkung an Hand weiterer Eigenschaften vorgenommen. Die einzelne Rekonfiguration eines Knotens und des mit ihm assoziierten Teilgraphen, die auf einen spontanen Entschluss des Fachanwenders zurückgeht, entspricht der Ad-hoc-Änderung eines Prozesses durch manuelle Abweichungsmaßnahmen. Prozessvarianten hingegen können als eine vordefinierte Menge von Schritten zur Konfiguration bzw. Rekonfiguration eines SPF-Graphen betrachtet werden.

Nach der Rekonfiguration des SPF-Graphen müssen die Maßnahmen in entsprechende Änderungen an der Prozessdefinition übersetzt werden. Dies geschieht, indem erneut eine Transformation des SPF-Graphen hin zu der Prozessdefinition durchgeführt wird. Damit nicht der gesamte SPF-Graph zum wiederholten Male umgewandelt werden muss, erstreckt sich die Transformation lediglich auf den minimalen Block aus Prozessfragmenten, der durch die Blattknoten des modifizierten Teilgraphen definiert ist. Bei der dynamischen Adaption einer Prozessinstanz ergeben sich zusätzliche Einschränkungen mit Hinblick auf den aktuellen Ausführungsstatus. So dürfen bereits durchgeführte Prozessaktivitäten oder Aktivitäten, die sich aktuell in Ausführung befinden, nicht von der Transformation betroffen sein. Um sicherzustellen, dass die Änderung der Prozessinstanz ihre weitere Ausführbarkeit nicht gefährdet, ist die exakte Spezifikation der Zustandssemantik von großer Bedeutung. In ADEPT2 sind die einzelnen Zustände der Prozessaktivitäten, mögliche Zustandsübergänge und die Folgen für den Status der Prozessinstanz genau definiert. Diese Angaben dienen im Rahmen des Lösungskonzeptes daher als Grundlage, um die korrekte Überführung von Prozessinstanzen nach der Rekonfiguration des SPF-Graphen zu ermöglichen. In den folgenden Kapiteln werden die einzelnen Phasen des Lösungskonzeptes detailliert beschrieben.

4.3 Definitionsphase

Ziel der Definitionsphase ist die Abbildung des prozessorientierten Domänenwissens in einem formal spezifizierten Modell. Auf diese Weise soll die Wiederverwendbarkeit einmal erstellter Prozessfragmente im Kontext unterschiedlicher Prozessdefinitionen sichergestellt werden. Darüber hinaus soll das Modell aber auch gewährleisten, dass das generelle Wissen, das während der Erhebung spezifischer Prozessabläufe gesammelt wird, nicht verloren geht, sondern verstetigt werden kann. Nur dann ist es möglich, dass die Prozessmodellierung Schritt für Schritt mit jeder entwickelten Prozessdefinition einfacher wird.

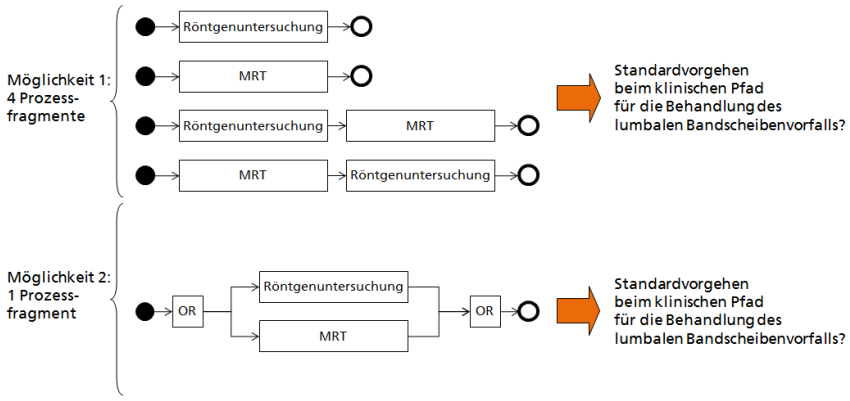
4.3.1 Motivation für ein Domänenmodell als Basis klinischer Pfade

Die meisten Modellierungsumgebungen im Umfeld von WfMS zielen bereits auf die Möglichkeit der Wiederverwendung einmal erstellter Prozessteile und Anwendungskomponenten ab. So besteht die Option, wiederkehrende Prozessteile in eine separate Definition auszulagern und aus unterschiedlichen übergeordneten Prozessdefinitionen heraus zu referenzieren. Dasselbe gilt für Anwendungskomponenten, die z. B. im Fall des adaptiven WfMS ADEPT2 als Aktivitätsvorlagen in einem entsprechenden Repository verwaltet werden. In einigen Systemen können die Prozessmodellierer Prozessteile und Anwendungskomponenten auch mit terminologischen Begriffen verschlagworten, um die Suche nach den Bausteinen zu vereinfachen und damit die Wahrscheinlichkeit für ihre Wiederverwendbarkeit zu erhöhen.

Mit der Kennzeichnung von Prozessteilen als wieder verwendbare Bausteine und ihrer Verwaltung in einem Repository werden zwar die Grundlagen für das Management von Prozesswissen innerhalb einer Domäne gelegt, die Zusammenhänge zwischen den einzelnen Bausteinen werden auf diese Weise jedoch nicht deutlich. Selbst wenn durch Verschlagwortung Abhängigkeiten und Verknüpfungen signalisiert werden, basieren diese Maßnahmen nicht auf einem formal definierten Mechanismus und sind mit dem Anwachsen des Prozesswissens in absehbarer Zeit nicht mehr beherrschbar. Mit der Zunahme der Bedeutung des Prozessmanagements und der Anzahl an Prozessdefinitionen, steigt daher auch der Bedarf an Möglichkeiten, das Prozesswissen in einem übergeordneten Modell abbilden zu können und mit wieder verwendbaren Prozessfragmenten zu assoziieren. Wie in Kapitel 4.7 gezeigt wird, gilt dasselbe auch für Varianten vom Prozessstandard, die sich gerade im klinischen Umfeld häufig nicht speziell für einen Prozess gelten, sondern in vielen unterschiedlichen klinischen Pfaden von Bedeutung sind.

Diese Problemsstellung soll an Hand eines Beispiels verdeutlicht werden: Im Rahmen der Diagnostik von Wirbelsäulenerkrankungen werden eine Reihe radiologischer Untersuchungen durchgeführt. Radiologische Untersuchungen gehören aber auch zur Basis- und erweiterten Diagnostik in vielen anderen Behandlungsszenarios. Auf Grund des hohen Grades an Wiederverwendbarkeit soll also ein Prozessfragment spezifiziert werden, das die Durchführung aller innerhalb einer Klinik angebotenen radiologischer Leistungen umfasst. Je nach Behandlungsszenario kann die Art und Reihenfolge der durchzuführenden Untersuchungen jedoch variieren. Das bedeutet, dass für unterschiedliche klinische Pfade unterschiedliche Versionen des Fragments benötigt werden. Gemäß dem traditionellen Vorgehen zu Verwaltung wieder verwendbarer Prozessfragmente sind zwei Möglichkeiten denkbar: Im ersten Fall werden genauso viele Varianten des Prozessfragments modelliert, wie benötigt werden; angesichts der Vielfalt an Untersuchungsmethoden und der resultierenden Bandbreite an Kombinationsmöglichkeiten stellt dies einen hohen Modellierungsaufwand dar, auch wenn vom Standard abweichende Behandlungsfälle noch gar nicht berücksichtigt werden. Darüber hinaus wird durch die enorme Menge an fast identischen Prozessfragmenten ihre Selektion im Zuge der Wiederverwendung erschwert. Alternativ können alle Varianten innerhalb eines einzigen Prozessfragments ausmodelliert werden; dasselbe Fragment kann dann in beliebigen Prozessdefinitionen Verwendung finden. Hierbei ergibt sich allerdings das Problem, dass es nicht möglich ist, Behandlungsstandards für die unterschiedlichen klinischen Pfade festzulegen. Gehören z. B. im Fall des lumbalen Bandscheibenvorfalls eine Röntgenuntersuchung und ein MRT zur Basisdiagnostik, reicht bei einer Fraktur standardmäßig eine Röntgenuntersuchung aus. Diese beispielhafte Problemstellung wird in Abbildung 29 illustriert.

Abbildung 29 Problematik beim Management alternativer Prozessfragmente



Da also übliche Vorgehensweisen bei der Verwaltung wieder verwendbarer Prozessfragmente an ihre Grenzen stoßen, adressieren neuere Ansätze des Workflow Managements die Problematik mit Hilfe der Prozesskonfiguration. Entsprechende Lösungen wurden bereits auch in Kapitel 3.3.3 und in Kapitel 3.3.4 vorgestellt. Diese Konzepte sind dafür gedacht, die Variabilität im Hinblick auf einen einzigen Geschäftsprozess abzubilden. Sie sind jedoch nicht dafür geeignet, das gesamte Prozesswissen innerhalb einer Domäne darzustellen. Beiden beschriebenen Ansätzen ist gemein, dass sie sich an einem speziellen Prozess, dem Basisprozess orientieren. Gemäß [Hallerbach et al. 2010] kann der Basisprozess dabei unterschiedliche Formen annehmen:

1. Der Basisprozess repräsentiert den Standard bei der Durchführung eines Prozesses z. B. zur Behandlung einer bestimmten Diagnose.
2. Der Basisprozess entspricht dem am häufigsten durchlaufenden Prozess z. B. zur Behandlung einer bestimmten Diagnose.
3. Der Basisprozess wird durch die Bestimmung des durchschnittlich minimalen Abstands zwischen sämtlichen variablen Prozessabläufen gebildet.
4. Beim Basisprozess handelt es sich um die Gesamtheit aller Varianten, die über bedingte Verzweigungen wahlweise zur Verfügung gestellt werden.
5. Der Basisprozess wird als Schnittmenge der Prozesselemente aus allen verfügbaren Varianten konstruiert.

Während bei den Alternativen drei und fünf bezweifelt werden muss, ob dabei überhaupt noch ein strukturell korrekter und semantisch sinnvoller Prozess entstehen kann, führt Möglichkeit vier zu einer kaum mehr beherrschbaren Komplexität, selbst wenn nur die Variabilität eines speziellen Prozesses betrachtet wird (siehe Kapitel 3.3.1); für die Abbildung des gesamten Prozesswissens einer Domäne ist der Ansatz daher keineswegs geeignet. Die Alternativen eins und zwei wiederum entsprechen selbst einem speziellen Fall, der nur unter einem definierten Aspekt, wie der Behandlung einer bestimmten Diagnose oder der Durchführung einer konkreten Prozedur erstellt werden kann. Die Modellierung des prozessorientierten Domänenwissens in Form eines Basisprozesses und dessen Varianten entspricht daher keiner adäquaten Methodik. Dies ist auch nicht die Zielstellung der beschriebenen Ansätze. Um also einen möglichst hohen Grad an Wiederverwendbarkeit existierender Prozessfragmente zur Modellierung einer Vielzahl von Prozessen innerhalb einer Domäne zu erreichen, bleibt die Option, Prozessfragmente möglichst feingranular zu definieren. Damit die Selektion und Komposition der Fragmente auch für Fachanwender ohne fundiertes technisches Wissen möglich ist, müssen semantische Zusammenhänge und Abhängigkeiten zwischen den Fragmenten in einem eigenen, übergeordneten Modell definiert sein.

4.3.2 Feature-Modellierung zur Abbildung des prozessorientierten Domänenwissens

Der Bedarf für ein übergeordnetes Modell zur Abbildung der Variabilität und der Korrelationen von Geschäftsprozessen wurde bereits im Rahmen der Entwicklung von ERP-Systemen (Enterprise Ressource Planning Systems) identifiziert [Soffer et al. 2003]. Allerdings beschränken sich die Autoren bisher auf die Festlegung generischer Schritte zur Erstellung eines solchen Modells, die von konkreten Modellierungssprachen unabhängig sind; die Vorgehensweise wird dann am Beispiel von OPM (Object Process Methodology) [Dori 2002] evaluiert. Es werden jedoch weder Korrektheitskriterien aufgestellt, die die Konfigurierbarkeit des Modells über das Prozesswissen innerhalb einer Domäne sicherstellen, noch werden formale Methoden definiert, die die Ableitung konkreter Geschäftsprozesse auf Basis dieses Modells ermöglichen. Auch im Bereich der Produktentwicklung wurde die Notwendigkeit zur Abbildung von Variabilität in einem übergreifenden Modell schon lange erkannt (siehe Kapitel 3.3.3). Auf Grund der erhobenen Bedarfslage wurden unterschiedliche Vorgehensweisen zur Modellierung von Produktfamilien geschaffen; eine Produktfamilie repräsentiert eine Menge von Komponenten, mit denen die Bedürfnisse eines spezifischen Marktsegments erfüllt werden sollen. Die Entwicklung der Produktfamilie basiert also auf einer sorgfältigen Erhebung der Lösungsbausteine innerhalb der adressierten Domäne unter Einbezug vielfältiger Ausprägungsmöglichkeiten des resultierenden Produkts.

Eine der häufigsten Vorgehensweisen zur Abbildung von Produktfamilien ist die Feature-Modellierung. Die Feature-Modellierungsmethodik wurde ursprünglich von [Kang et al. 1990] vorgestellt und seitdem in einer Reihe von Veröffentlichungen weiterentwickelt und verfeinert [Kang et al. 1998, van Gurp et al. 2001, Batory 2005, van Deursen & Klint 2002, Riebisch et al. 2002, Riebisch 2003, Czarnecki et al. 2004, Czarnecki et al. 2005, Benavides et al. 2005]. Auch im Umfeld der Vorgehensweisen und Werkzeug-Entwicklung ergaben sich seit der ersten Erwähnung von Feature-Modellen große Fortschritte. Feature-Modellierung findet Anwendung in verschiedensten Bereichen und Branchen und dient u. a. den folgenden Zwecken [Czarnecki & Kim 2005]:

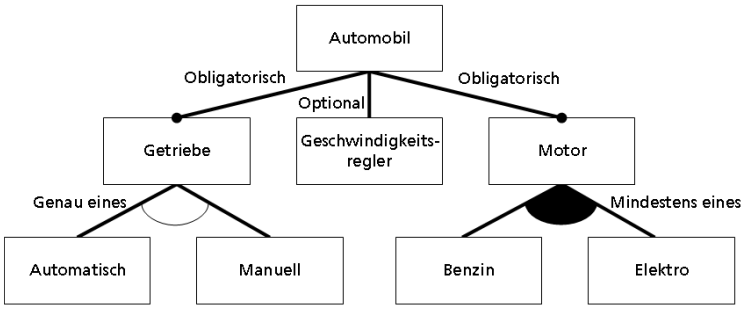
- Abbildung der Ergebnisse der Analyse einer Domäne / eines Anwendungsbereiches
- Organisation von Produktfamilien, domänen-spezifischen Sprachen, Anwendungskomponenten, Plattformen und anderen wieder verwendbaren Bausteinen
- Basis für die (automatisierte) Konfiguration konkreter Produkte, Sprachen, Anwendungskomponenten, Plattformen, usw.

Insgesamt kann festgestellt werden, dass das Ausdrücken von Variabilität von Produktfamilien auf Basis der Feature-Modellierungsmethodik intuitiver ist und weiter verbreitet als der von Soffer et al. vorgeschlagene Ansatz und die Modellierungssprache OPM. Aus diesem Grund soll das Vorgehen bei der Feature-Modellierung nun detaillierter vorgestellt werden und Rückschlüsse auf seine Eignung zur Entwicklung von Modellen über das domänenorientierte Prozesswissen gezogen werden.

Gemäß der in FODA (Feature-Oriented Domain Analysis) [Kang et al. 1990] und FORM (Feature-Oriented Reuse Method) [Kang et al. 1998] beschriebenen Methodik bestehen Feature-Modelle aus einem Feature Diagramm, das um zusätzliche Informationen, wie Beschreibungen zu Features, Prioritäten, Akteure, etc. erweitert ist. Feature Diagramme entsprechen einer graphischen, baumartigen Organisation singulärer Features. Jedes Feature repräsentiert dabei eine aus Sicht des Anwenders besondere Ausprägung bzw. Eigenschaft des zu kreierenden Produktes; bei Features für Softwareprodukte innerhalb von Automobilen kann es sich hierbei z. B. um die Art des Getriebes (automatisch oder manuell) oder die Option auf eine

Geschwindigkeitsregelungsanlage handeln. Ausgehend von der Grundidee einer graphischen Repräsentation von Feature-Modellen, wurden in den vergangenen Jahren alternative, graphische Repräsentationsformen für Feature-Modelle geschaffen. Abbildung 30 illustriert ein Beispiel für eine mögliche Repräsentationsform.

Abbildung 30 Beispiel für die graphische Repräsentation eines Feature-Modells [Benavides et al. 2006]



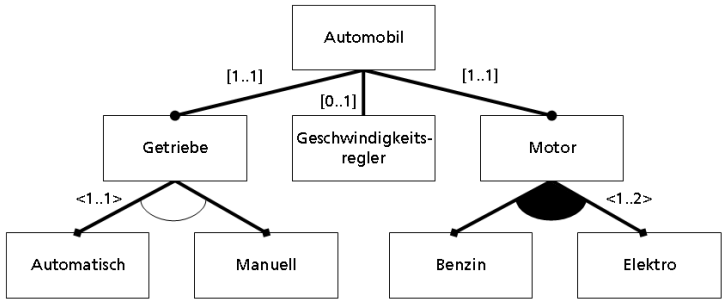
Nach Kang et al. entspricht der Wurzelknoten eines Feature Diagramms einem Konzeptknoten, wobei die Bezeichnung der Produktklasse, die abgebildet werden soll, stellvertretend für das Konzept stehen kann; bei allen Knoten unterhalb der Wurzel handelt es sich um Features. Die Autoren klassifizieren Features als verpflichtend, optional und alternativ; die baumartige Strukturierung ist dementsprechend um Informationen zu logischen Verknüpfungen angereichert, die Aussagen über die Beziehungen der Kindknoten eines übergeordneten Elternknotens machen. In dem Beispiel aus obiger Abbildung bilden die Features »Automatisch« und »Manuell« unterhalb des Elternknotens »Getriebe« Alternativen zueinander. Der Vorgang der Selektion von Features zur Charakterisierung eines spezifischen Produktes wird auch als Konfiguration bezeichnet. Eine mögliche Konfiguration für ein Automobil gemäß Abbildung 30 verfügt demnach über die Features »Getriebe«, »Automatisch«, »Geschwindigkeitsregler«, »Motor«, »Benzin« und »Elektro«. Es gibt in der Literatur keinen Konsens darüber, welche Arten von logischen Verknüpfungen bereit stehen sollten, um eine Menge von Knoten zueinander in Beziehung zu setzen. Die folgenden Operatoren werden am häufigsten für die Verwendung in Feature-Modellen genannt [Sun et al. 2005]:

- Verpflichtend: Alle Features, auf die sich dieser Operator bezieht, müssen bei der Selektion des übergeordneten Features ebenfalls gesetzt werden.
- Optional: Alle Features können optional ausgewählt werden.
- Alternativ: Die folgenden Features können nur alternativ zueinander gewählt werden.
- Mindestens eines: Von den folgenden Features muss mindestens eines selektiert sein
- Optionale Alternative: Von den folgenden Features darf entweder genau eines oder keines ausgewählt werden
- Beliebig: Von den folgenden Features können beliebig viele oder auch keines gesetzt werden

Anzumerken ist, dass die beiden letzten Operatoren auch durch Kombinationen der Operatoren »Optional« und »Alternativ« bzw. »Optional« und »Mindestens eines« gebildet werden können. Eine andere Möglichkeit, um die Verpflichtung bzw. die Option zum Einschluss eines Features in eine Produktkonfiguration auszudrücken, besteht in der Angabe von Kardinalitäten zu Features [Czarnecki et al. 2004]. Optionale Features erhalten die Kardinalität [0..1], verpflichtenden Features wird hingegen die Kardinalität [1..1] zugewiesen. Um zu signalisieren, dass eine Menge von Features alternativ gewählt werden kann, werden Features zu so ge-

nannten »Feature Gruppen« zusammengefasst. Die Kardinalität gilt dann nicht für das einzelne Feature, sondern für die gesamte Gruppe; d. h. mit Hilfe der Gruppenkardinalität wird spezifiziert, wie viele Features aus einer Gruppe in einer Konfiguration gesetzt sein dürfen. Die folgende Abbildung zeigt das kardinalitätsbasierte Feature-Modell für das »Automobil«-Beispiel.

Abbildung 31 Beispiel für die graphische Repräsentation eines kardinalitätsbasierten Feature-Modells



Während es sich in der obigen Darstellung bei den Features »Getriebe«, »Geschwindigkeitsregler« und »Motor« um singuläre Features handelt, repräsentieren »Automatisch« und »Manuell« bzw. »Benzin« und »Elektro« jeweils Features innerhalb einer Feature Gruppe. Die Annotation von Kardinalitäten ermöglicht darüber hinaus auch das Klonen singulärer Features. Das bedeutet, dass identische Features mehr als einmal in derselben Konfiguration vorkommen können; verfügt das jeweilige Feature über weitere untergeordnete Features, kann dabei die Konfiguration der Kindknoten des ursprünglichen Features von der Konfiguration der Kindknoten des geklonten Features abweichen.

Neben der hierarchischen Strukturierung von Features ist in den meisten Ansätzen in Zusammenhang mit Feature-Modellierung noch ein weiteres Mittel zur Darstellung von Beziehungen zwischen Features vorgesehen: Constraints repräsentieren Aussagen, die für jede Konfiguration eines Feature-Modells erfüllt sein müssen. Bei den zwei am häufigsten erwähnten Constraints handelt es sich um »Anforderung« und »Ausschluss«. Mit Hilfe der Constraints können Beziehungen zwischen zwei Features auch auf einer horizontalen Achse definiert werden. Der Ausdruck »Feature *A* erfordert Feature *B*« bedeutet, dass jede Konfiguration des Feature-Modells, die das Feature *A* enthält, auch das Feature *B* umfassen muss. »Feature *A* schließt Feature *B* aus« soll hingegen bewirken, dass eine Konfiguration, die Feature *A* enthält, Feature *B* nicht einschließen darf.

Wie in den folgenden Kapiteln gezeigt wird, eignet sich die Methodik der Feature-Modellierung auch zur Abbildung des Prozesswissens innerhalb einer Domäne. Analog zu Produktfamilien, werden im Fall des Prozesswissens Prozessfamilien und ihre Variabilität in Form eines Feature-Modells beschrieben. Dabei repräsentiert das einzelne Feature aus Sicht des Anwenders bzw. Prozessmodellierers eine besondere Ausprägung, die mindestens im Kontext eines Geschäftsprozesses von Bedeutung ist. Bei einem Feature kann es sich beispielsweise um eine radiologische Untersuchung handeln, deren untergeordnete Features sich auf die verschiedenen Methoden der Bild gebenden Diagnostik innerhalb eines Krankenhauses beziehen. Auch aus Sicht der Vorgehensweise kommt das Konzept der Feature-Modellierung den Anforderungen bei der Abbildung des prozessübergreifenden Wissens innerhalb einer Domäne entgegen. Während sich die traditionelle Prozessmodellierung auf den einzelnen Geschäftsprozess, seine Aktivitäten, Schnittstellen und Akteure fokussiert, gelingt es mit der Feature-

Modellierung von dem singulären Geschäftsprozess zu abstrahieren und sich stattdessen zunächst auf die Leistungsbereiche eines Krankenhauses zu konzentrieren. Im Sinne eines »Bottom-up« Verfahrens können die Leistungen einzelner Stationen bzw. Abteilungen oder Personengruppen spezifiziert werden, die letztlich zur Erbringung einer ganzen Reihe von Geschäftsprozessen beitragen. Im Gegensatz zu den Details eines Geschäftsprozesses ist das Leistungsangebot eines Krankenhauses genau definiert, da für die Abrechnung mit den Kostenträgern die Zuordnung zu den bundesweit gültigen Leistungskatalogen erforderlich ist[34]; darüber hinaus existieren häufig klinikinterne Leistungskataloge, die für die Identifikation von Features genutzt werden können. Schließlich müssen die Tätigkeiten bestimmt werden, die zur Erbringung einer Leistung notwendig sind; danach können in Abhängigkeit der Tätigkeit (z. B. Visite als manuelle und mobile Aktivität im Gegensatz zur automatischen Auftragskommunikation) Prozessfragmente entwickelt werden. Im Gegensatz zur traditionellen Erstellung von Prozessdefinitionen sollten Beziehungen und Ablaufreihenfolgen zwischen den als Feature-Modellierten Leistungen und Tätigkeiten nur optional spezifiziert werden und zwar dann, wenn diese grundsätzlich Gültigkeit besitzen sollen (z. B. bei etablierten und bewährten Routineabläufen oder auf Grund von Datenabhängigkeiten).

Obwohl Kang et al. bereits in ihrer ersten Veröffentlichung zu Feature-Modellen die Notwendigkeit einer formalen Grundlage betonen, entbehren die meisten Lösungsansätze bis heute einem solchen Formalismus. Der Bedarf für die Formalisierung von Feature-Modellen wurde auch im Endbericht des ECOOP (Workshops on Modeling Variability for Object-Oriented Product Lines) hervorgehoben [Riebisch et al. 2004]. Erst in den letzten Jahren haben einige Autoren begonnen, eine formale Semantik für die existierenden Konzepte und ihre Erweiterungen zu definieren [Mannion 2002, Czarnecki et al. 2005, Sun et al. 2006, Schobbens et al. 2006]. Diese Arbeiten sollen bei der formalen Spezifikation eines Modells zur Abbildung des Prozesswissens innerhalb einer Domäne berücksichtigt werden.

4.3.3 SPF-Typ-Graphen als Domänenmodelle für klinische Pfade

Obgleich sich die Feature-Modellierung generell für die Abbildung des Prozesswissens einer Domäne eignet, ist es notwendig, die Darstellungsvielfalt von Feature-Modellen auf die Konzepte zu beschränken, die tatsächlich von Nutzen sind. Darüber hinaus müssen Erweiterungen vorgenommen werden, die es möglich machen, konfigurierte Modelle letztlich in gültige Prozessdefinitionen zu transformieren und existierende Konfigurationen bei Bedarf flexibel zu ändern. In Abgrenzung zur konventionellen Feature-Modellierung und der Nutzung von Feature-Modellen zur Darstellung der Variabilität innerhalb einer Produktfamilie wird daher an dieser Stelle der Begriff des »SPF-Typ-Graphen« eingeführt, der speziell auf die Abbildung des Prozesswissens innerhalb einer Domäne ausgerichtet ist.

[34] Vgl. Deutsches Institut für Medizinische Dokumentation und Information, DIMDI (Hrsg. im Auftrag des Bundesministeriums für Gesundheit unter Beteiligung der Arbeitsgruppe OPS des Kuratoriums für Fragen der Klassifikation im Gesundheitswesen (KKG): OPS Version 2009 Systematisches Verzeichnis, Operationen- und Prozedurenschlüssel, Internationale Klassifikation der Prozeduren in der Medizin (OPS), Stand: 20.10.2008, www.dimdi.de

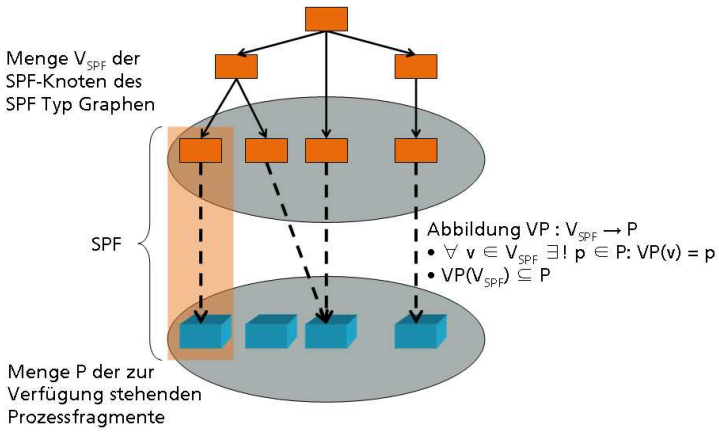
Definition (SPF-Typ-Graph). Ein SPF-Typ-Graph ist ein azyklischer, gerichteter Graph zur Abbildung und Organisation des Prozesswissens innerhalb einer Domäne. Jeder SPF-Typ-Graph hat mindestens einen ausgezeichneten Startknoten (Wurzel). Alle Knoten, die nicht über gerichtete Kanten mit nachfolgenden Knoten verbunden sind, werden als »SPF-Knoten« bezeichnet. Alle Knoten innerhalb des SPF-Typ-Graphen, die keine SPF-Knoten sind, erhalten die Bezeichnung »Kontextknoten«.

Wie in Kapitel 4.2 erstmals festgelegt wurde, steht das Kürzel »SPF« für »Semantisches Prozessfragment«. Prozessfragmente repräsentieren ausführbare Einheiten, die über eine Schnittstellenspezifikation und eine Implementierung verfügen und als elementare Bausteine zur Modellierung von Prozessdefinitionen eingesetzt werden können. In Kapitel 4.3.4 werden Prozessfragmente formal spezifiziert.

Definition (Semantisches Prozessfragment, kurz: SPF). Ein Semantisches Prozessfragment ist ein Prozessfragment, das einem SPF-Knoten innerhalb eines SPF-Typ-Graphen zugeordnet wurde.

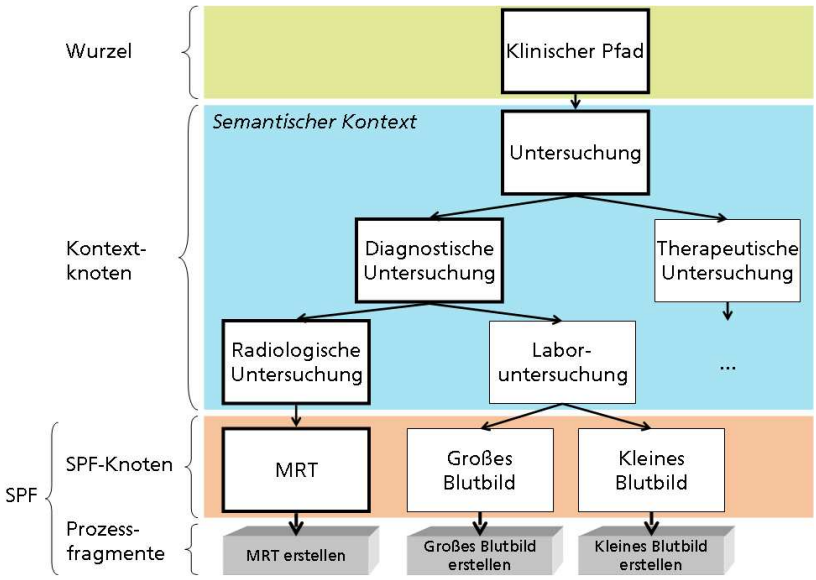
Durch diese Definition des SPFs ergibt sich eine direkte Anforderung an SPF-Typ-Graphen, die für Feature-Modelle nicht gilt. SPF-Typ-Graphen müssen über eine Funktion verfügen, die die SPF-Knoten des Graphen auf Prozessfragmente abbildet. Die nachfolgende Graphik illustriert den Zusammenhang zwischen den SPF-Knoten eines SPF-Typ-Graphen und der Menge der Prozessfragmente.

Abbildung 32 Bildung von SPFs durch die Zuordnung von elementaren Knoten des SPF-Typ-Graphen zu Prozessfragmenten



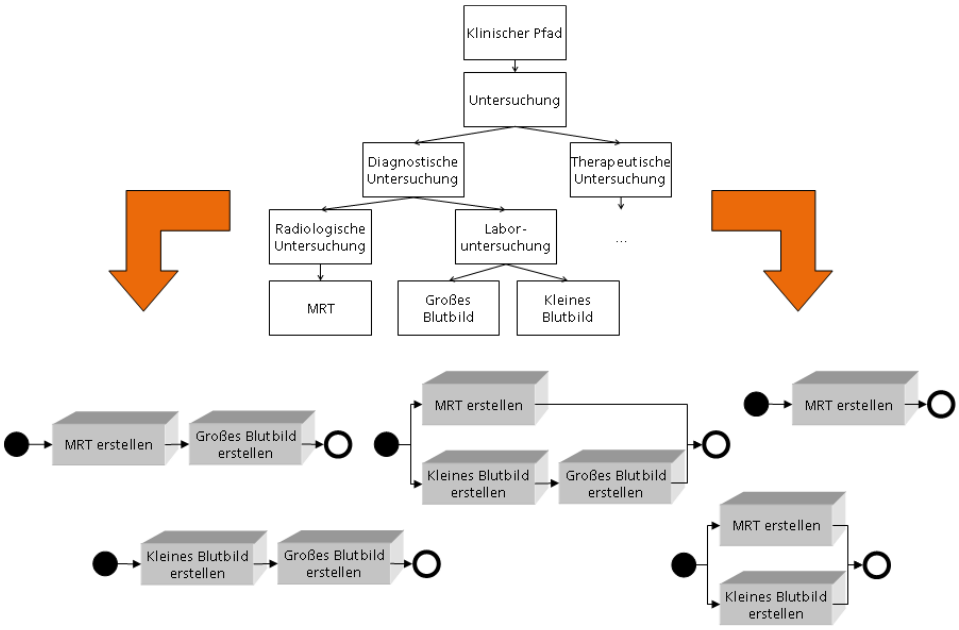
Wie aus der Abbildung ersichtlich ist, wird jedem SPF-Knoten des SPF-Typ-Graphen mittels der Abbildungsfunktion $VP: V_{SPF} \nrightarrow P$ genau ein Prozessfragment zugewiesen. Dabei können sich auch unterschiedliche Knoten auf dasselbe Prozessfragment beziehen. Wieder andere Prozessfragmente werden von keinem SPF-Knoten referenziert und bilden daher auch kein SPF in Zusammenhang mit dem betrachteten SPF-Typ-Graphen. Analog zu Feature Diagrammen wird mittels gerichteter Kanten im SPF-Typ-Graphen eine Knotenhierarchie konstruiert. Diese Knotenhierarchie definiert für jedes SPF einen semantischen Kontext, innerhalb dem das SPF selektiert werden kann (siehe Abbildung 33).

Abbildung 33 Bildung eines semantischen Kontexts durch die Knotenhierarchie des SPF-Typ-Graphen



Solange SPF-Typ-Graphen lediglich eine hierarchische Struktur auf Prozessfragmente definieren, ist der Grad an Unterstützung für Fachanwender bei der Prozessmodellierung noch sehr gering. Im Vorgriff auf Kapitel 4.5, in dem der eigentliche Transformationsprozess, d. h. die Gewinnung der Prozessdefinition auf Basis der Konfiguration des SPF-Typ-Graphen, dargestellt wird, soll daher nun informal untersucht werden, inwiefern die Prozessmodellierung auf Basis des bisher beschriebenen Konzepts beeinflusst werden kann und speziell welche Erleichterungen sich für die Fachanwender ergeben. Ausgehend von dem Beispiel in Abbildung 33 zeigt die nachfolgende Graphik, welche Variantenvielfalt an möglichen Prozessdefinitionen auf Basis der aktuellen Festlegungen in dem SPF-Typ-Graphen erzeugt werden kann, sobald sich der Anwender entscheidet, dass eine diagnostische Untersuchung durchgeführt werden soll; dabei sei angemerkt, dass nur ein kleiner Teil aller möglichen Alternativen in der Abbildung enthalten sind. Durch die Auswahl des Kontextknotens »Diagnostische Untersuchung« stehen dem Fachanwender gemäß dem Beispiel die Prozessfragmente »MRT erstellen«, »Großes Blutbild erstellen« und »Kleines Blutbild erstellen« zur Verfügung, die er nun in beliebiger Menge und in beliebiger Reihenfolge anordnen kann, da der SPF-Typ-Graph diesbezüglich keine eindeutigen Aussagen macht.

Abbildung 34 Deklarative Prozessmodellierung auf Basis von SPF-Typ-Graphen

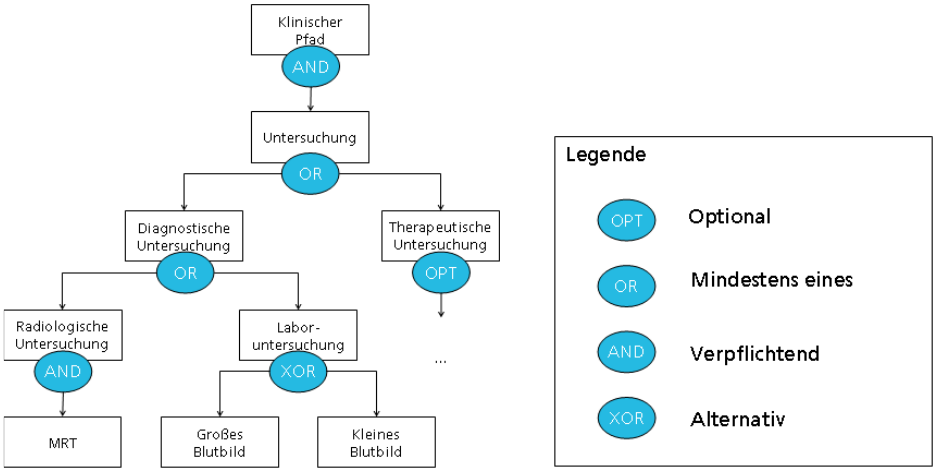


Aus diesem Beispiel muss der Schluss gezogen werden, dass der Aufwand für die Anwender auf der Grundlage der bisherigen konzeptionellen Beschreibungen im Hinblick auf SPF-Typ-Graphen hoch bleibt. Durch die Selektion eines übergeordneten Kontextknotens wird zwar eine Vorauswahl der in Frage kommenden Prozessfragmente getroffen; allerdings erhält der Anwender keinerlei Unterstützung bei der Komposition dieser Fragmente zu kompletten Prozessdefinitionen. Die Art und Weise der Prozessmodellierung auf Grundlage der bisher in einem SPF-Typ-Graphen verfügbaren Informationen weist starke Parallelen zu der deklarativen Prozessmodellierung auf (siehe Kapitel 3.2.4). Analog dazu schreibt der SPF-Typ-Graph in seiner aktuellen Form vor, welche Prozessfragmente zur Verfügung stehen und erweitert diese Information um die Aussage, in welchem Kontext ein Prozessfragment in eine Prozessdefinition eingebettet wird. Allerdings bleibt es dem Prozessmodellierer selbst überlassen, welche der Fragmente er tatsächlich nutzen möchte und in welcher Reihenfolge er die Fragmente in der Prozessdefinition anordnet. Die Erhaltung eines gewissen Freiheitsgrades bei der Prozessmodellierung ist auch durchaus wünschenswert; dennoch gibt es in jedem Krankenhaus gut funktionierende Routineabläufe, die grundsätzlich zu berücksichtigen sind, es existieren Qualitätsrichtlinien oder SOLL-Prozesse, die in verstärktem Maße durchgesetzt werden sollen. In diesem Entwicklungsstatus bildet der SPF-Typ-Graph nur einen Teil des verfügbaren Prozesswissens innerhalb der Domäne ab, da noch keine Mittel bereitstehen, um solche generellen Abläufe in dem Graph zu verankern. Damit die Behandlungsprozesse in einem Krankenhaus möglichst effizient durchgeführt werden können, sind die Verstetigung der bereits existierenden und weiterhin erwünschten Routineabläufe sowie die konsequente Umsetzung neuer Regelungen im Rahmen der Definitionsphase jedoch unumgänglich. Aus diesem Grund muss der SPF-Typ-Graph auch darstellen, welche generellen Beziehungen und Abhängigkeiten zwischen Prozessfragmenten existieren, die während der Prozessmodellierung beachtet werden müssen.

Ein erstes Mittel, um SPF-Typ-Graphen um Informationen zu solchen Zusammenhängen zu erweitern, Prozessdefinitionen in dem gewünschten Maße stärker zu vereinheitlichen und den Modellierungsvorgang zu vereinfachen, besteht darin, festzulegen, ob bestimmte Knoten in-

nerhalb des Graphen obligatorisch, optional oder wahlweise zur Selektion bereitstehen. In Kapitel 4.3.2 zur Feature-Modellierung wurden zwei Verfahren vorgestellt, um solche Zusammenhänge auszudrücken. Die erste Methode basiert auf der Anreicherung des Feature Diagramms um logische Verknüpfungen, die zweite Variante auf der Angabe von Kardinalitäten. kardinalitätsbasierte Feature-Modelle haben den Nachteil, dass sie das Konstrukt der Featuregruppe benötigen, um Features darzustellen, die alternativ oder wahlweise selektiert werden können. Die Option, Features zu klonen besteht aber ausschließlich im Fall eines singulären Features; das bedeutet, sobald ein Knoten des SPF-Typ-Graphen Teil einer Gruppe ist, beträgt seine Kardinalität maximal eins. Aus diesem Grund sollen SPF-Typ-Graphen um eine Funktion verfügen, die dem Wurzel- und jedem Kontextknoten einen logischen Operator zuweist, der eine Aussage über die Zusammenstellung der nachfolgenden Knoten macht. Abbildung 35 demonstriert die Anreicherung des Beispiel-Graphen aus Abbildung 33 um logische Operatoren, wobei die Kürzel »OPT«, »OR«, »AND« und »XOR« jeweils den in Kapitel 4.3.2 vorgestellten Verknüpfungsarten entsprechen.

Abbildung 35 Zuordnung von logischen Verknüpfungen zu Knoten des SPF-Typ-Graphen



Die Konfiguration, also die Selektion von Knoten des SPF-Typ-Graphen, verläuft in dem Beispiel nach dem folgenden Schema:

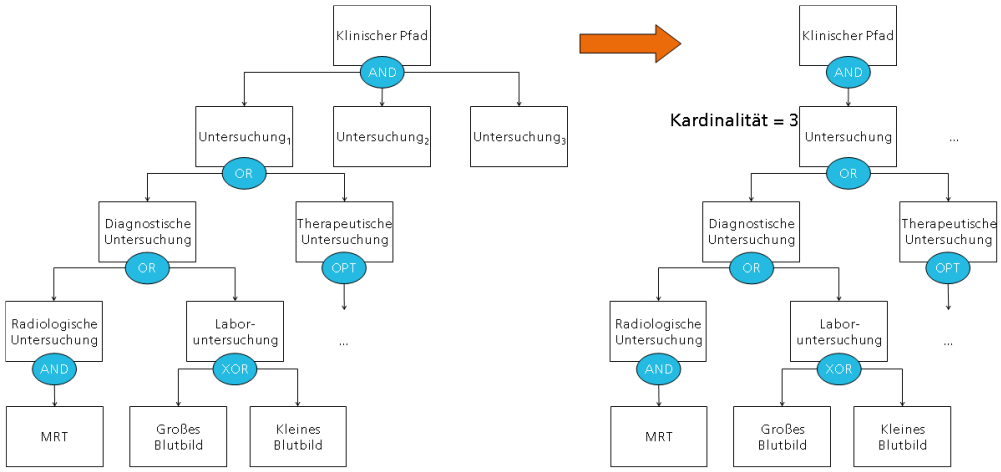
- Ausgehend von der Wurzel »Klinischer Pfad« muss der Knoten »Untersuchung« selektiert werden
- Wird der Knoten selektiert, muss anschließend mindestens einer der Knoten »Diagnostische Untersuchung« und »Therapeutische Untersuchung« gesetzt werden
- Die diagnostische Untersuchung besteht mindestens aus »Radiologische Untersuchung« oder »Laboruntersuchung«
- Bei Wahl der radiologischen Untersuchung ist automatisch der Knoten »MRT« zu setzen
- Bei der Laboruntersuchung muss zwischen den Alternativen »Großes Blutbild« und »Kleines Blutbild« gewählt werden

An diesem Beispiel kann man bereits erkennen, dass logische Verknüpfungen ein probates Mittel repräsentieren, um die Vielfalt möglicher Modellierungsformen auf ein geeignetes und beherrschbares Maß zu reduzieren. Gemäß diesem Verfahren kann es vorkommen, dass Knoten innerhalb des SPF-Typ-Graphen nur eingeführt werden, um Konfigurationsvorgaben für die nachfolgenden Knoten aufzustellen, z. B. bei der Kombination von »OPT« und »XOR« um eine optionale Alternative auszudrücken. Diese Knoten, die ausschließlich der Zuweisung

des logischen Operators dienen, können jedoch genauso behandelt werden wie normale Kontextknoten.

Wie bereits erwähnt, spielt die Angabe einer Kardinalität nicht nur beim Ausdrücken logischer Verknüpfungen zwischen den Knoten des SPF-Typ-Graphen eine Rolle, sondern kann auch genutzt werden, um zu signalisieren, wie viele Knoten dieses Typs in der Konfiguration des Graphen vorliegen dürfen. Ohne die Zuweisung einer Kardinalität basiert die Konfiguration auf der Annahme, dass jeder Knoten entweder maximal einmal oder beliebig oft selektiert werden darf. Da in der klinischen Domäne viele Aktivitäten wiederholt durchgeführt werden, würde der SPF-Typ-Graph eine unnötige Komplexität annehmen, wenn pro potentielle Wiederholung ein identischer Teilgraph gebildet werden müsste. Genauso wenig Sinn macht die unendliche Wiederholung einer Untersuchung oder einer Prozedur während des stationären Aufenthalts des Patienten. Aus diesem Grund wird jedem Knoten des SPF-Typ-Graphen eine maximale Kardinalität zugeordnet. In Abbildung 36 wird die sich dadurch ergebende Komplexitätsreduktion am Beispiel des Kontextknotens »Untersuchung« demonstriert.

Abbildung 36 Knotenkardinalitäten im SPF-Typ-Graphen

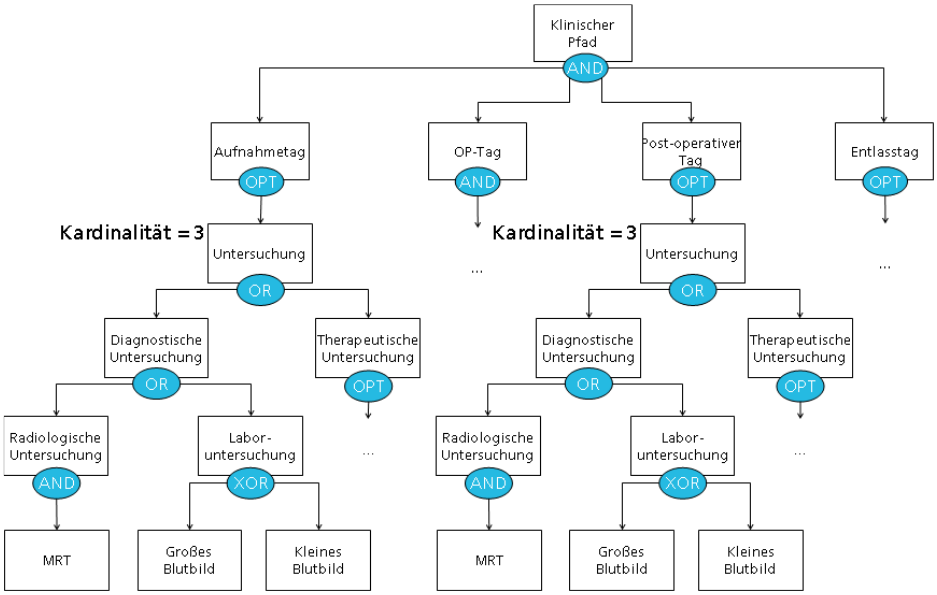


Im Gegensatz zu dem in [Czarnecki et al. 2004] vorgestellten Ansatz, können im Fall von SPF-Typ-Graphen für jeden Knoten eine minimale Untergrenze *min = 1* und eine maximale Obergrenze *max* $\in \mathbb{N} \setminus 0$ festgelegt werden. Grund für diese Entscheidung ist, dass das wiederholte Durchführen von Untersuchungen und Prozeduren in Krankenhäusern zum einen eher vermieden werden soll und dass klinische Pfade zum anderen eine abgeschlossene Behandlung beschreiben und keine unendliche Verweildauer des Patienten im Krankenhaus vorgesehen ist; demzufolge kann es auch keine Untersuchungen oder Prozeduren geben, die mit beliebiger Häufigkeit wiederholt werden dürfen. Darüber hinaus sollen auch keine Kardinalitäten nach dem Muster *[3..3][6..10]* unterstützt werden, die besagen, dass ein Feature entweder genau dreimal oder sechs bis zehnmal selektiert werden darf. Bei der Analyse klinischer Pfade hat sich bisher kein Bedarf für die Spezifikation solch komplexer Kardinalitäten ergeben. Dementsprechend wird die Kardinalität in SPF-Typ-Graphen nicht als Intervall, sondern lediglich mit der Angabe der maximalen Obergrenze festgesetzt.

Mit der Festsetzung von Knotenkardinalitäten lassen sich Klone lediglich unterhalb desselben Kontextknotens erzeugen. Es kann aber der Bedarf bestehen, dieselben Teilgraphen an unterschiedlichen Stellen des SPF-Typ-Graphen vorzusehen. Erfolgt z. B. eine zeitliche Struktu-

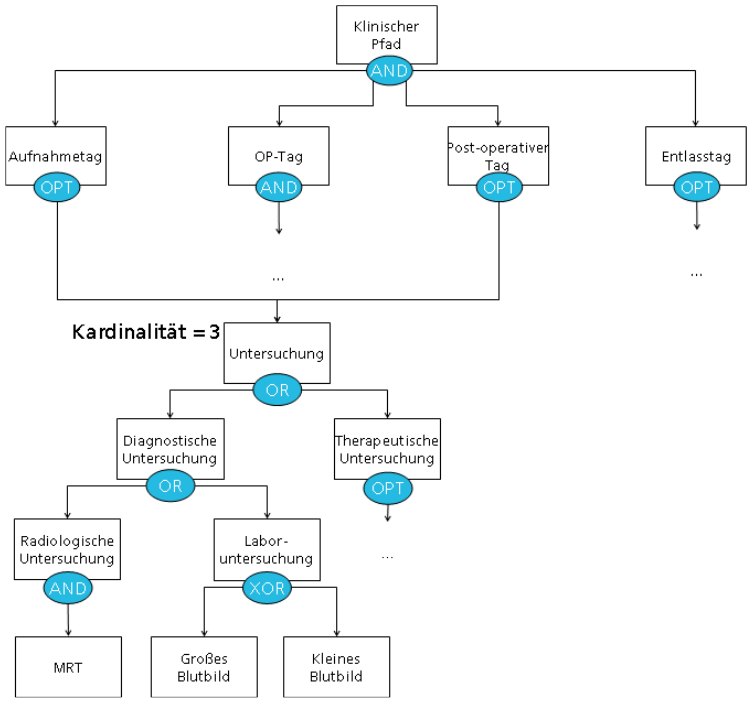
rierung des Graphen durch Einführung der Knoten »Aufnahmetag«, »OP-Tag«, »Post-operativer Tag« und »Entlasstag«, so ist es möglich, dass dieselben Untersuchungen an unterschiedlichen Tagen des klinischen Pfades stattfinden können. Gemäß dem aktuellen Konzept ist es dann notwendig, identische Teilgraphen jeweils mit unterschiedlichen Elternknoten zu spezifizieren (siehe Abbildung 37).

Abbildung 37 Identische Teilgraphen innerhalb eines SPF-Typ-Graphen



Um solche komplexen Strukturen zu vermeiden, ist es in SPF-Typ-Graphen möglich, von der strengen Baumstrukturierung abzuweichen und im Bedarfsfall »Mehrfachreferenzen« zu spezifizieren. Das bedeutet, dass derselbe Knoten von unterschiedlichen Kontextknoten als Nachfolger referenziert werden kann. Die folgende Abbildung zeigt, in welcher Weise der SPF-Typ-Graph durch die Möglichkeit der Mehrfachreferenz vereinfacht werden kann. Wird wie in der Abbildung geschehen, der Untersuchung die Kardinalität *drei* zugewiesen, bedeutet dies, dass sowohl am Aufnahme- als auch am OP-Tag jeweils maximal drei Untersuchungen stattfinden dürfen; die maximale Gesamtanzahl aller Untersuchungen beträgt demzufolge *sechs*.

Abbildung 38 Mehrfachreferenzen auf identische Teilgraphen innerhalb von SPF-Typ-Graphen



An dieser Stelle muss angemerkt werden, dass es immer noch durchaus sinnvoll und notwendig sein kann, identische Teilgraphen an unterschiedlichen Stellen im SPF-Typ-Graphen einzuführen; dies ist z. B. erforderlich, wenn vor der Durchführung einer Prozedur Laborwerte überprüft werden, die zu diesem Zweck speziell am OP-Tag gemacht werden müssen. In solchen Fällen muss es möglich sein, schon auf der Ebene des SPF-Typ-Graphen zwischen der Laboruntersuchung am OP-Tag und der Laboruntersuchung am Aufnahmetag zu unterscheiden. Aus Sicht des SPF-Typ-Graphen handelt es sich dann jedoch tatsächlich um zwei voneinander verschiedene Teilgraphen.

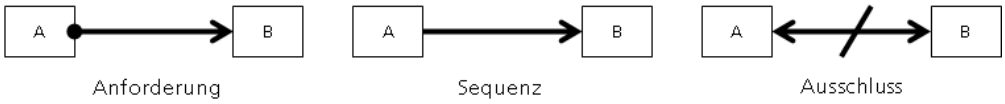
Da es mit logischen Verknüpfungen ausschließlich möglich ist Zusammenhänge zwischen Knoten mit demselben übergeordneten Kontextknoten auszudrücken, bedient sich sowohl die deklarative Prozessmodellierung als auch die Feature-Modellierung einem speziellen Mittel zur Definition genereller Beziehungen zwischen Knoten auf horizontaler Ebene: Mit Hilfe von Constraint-Relationen können grundlegende Wahrheiten formuliert werden, die innerhalb jeder Prozessdefinition gültig sind. Analog dazu soll auch für SPF-Typ-Graphen die Möglichkeit bestehen, durch Constraints Beziehungen zwischen Knoten auszudrücken. Insgesamt werden in dieser Arbeit drei Arten von Constraints unterschieden (siehe Abbildung 39):

- **Anforderung:** Der Ausdruck »Knoten *A erfordert* Knoten *B*« bedeutet, dass in einer gültigen Konfiguration des SPF-Typ-Graphen mindestens ein Knoten vom Typ *B* unterhalb des kleinsten gemeinsamen Kontextknoten von *A* und *B* existieren muss, sobald es einen Knoten vom Typ *A* gibt. Weiterhin folgt, dass in der entsprechenden Prozessdefinition, vor allen Prozessfragmenten, die *A* oder einem Nachfolger von *A* zugeordnet sind, alle Prozessfragmente, die *B* oder einem Nachfolger von *B* zugeordnet sind, ausgeführt werden müssen; dies gilt auch für alle Klone von *A* und *B*.
- **Sequenz:** Der Ausdruck »Knoten *A folgt auf* Knoten *B*« bewirkt, dass alle Prozessfragmente, die *A* oder einem Nachfolger von *A* zugeordnet sind, hinter allen Prozessfragmenten, die

B oder einem Nachfolger von *B* zugeordnet sind, ausgeführt werden sollen, sofern diese in der Prozessdefinition existieren; dies gilt analog für deren Klone.

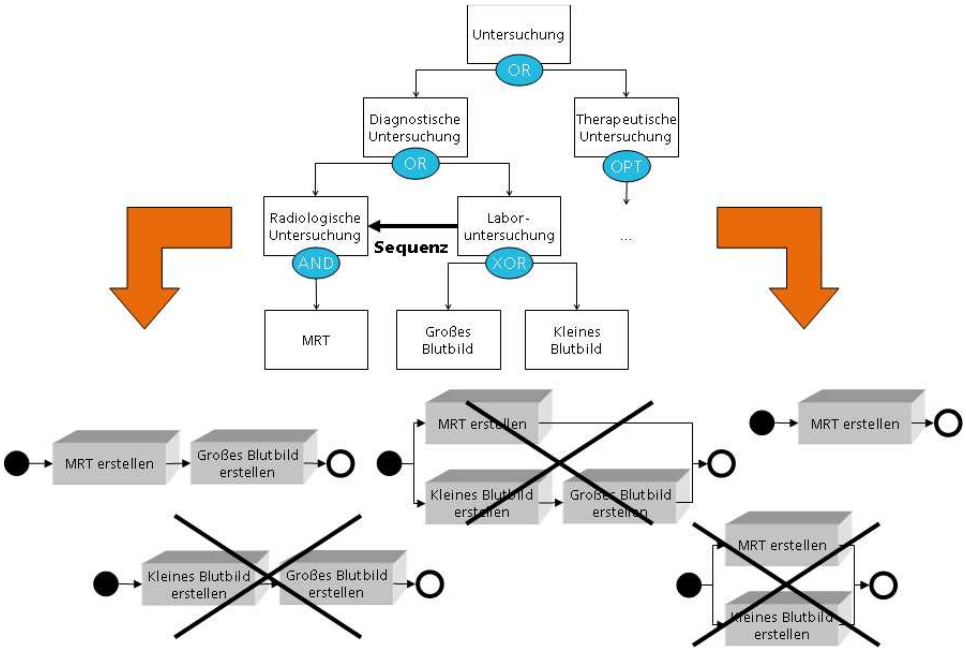
- **Ausschluss:** Der Ausdruck »Knoten *A schließt* Knoten *B aus*« bedeutet, dass in jedem Teilgraph der Konfiguration eines SPF-Typ-Graphen, der ausgehend von dem kleinsten gemeinsamen, übergeordneten Knoten von *A* und *B* gebildet wird, nur entweder *A* oder *B* einschließlich deren Klone existieren dürfen. Es handelt sich hierbei also um einen gegenseitigen Ausschluss.

Abbildung 39 Constraint-Relationen in SPF-Typ-Graphen



Die Festlegung dieser drei Constraints basiert nicht nur auf der Analyse der Anforderungen an Prozessdefinitionen im klinischen Bereich, sondern hat seinen Hintergrund auch in den unterschiedlichen Auswirkungen der Constraints während der Konfigurations-, Transformations- und Rekonfigurationsphase. Wie in den Kapiteln zu den unterschiedlichen Phasen ausführlich dargestellt wird, hat der Anforderungs-Constraint nicht nur Einfluss auf die Gestaltung von SPF-Graphen während der Konfiguration bzw. Rekonfiguration, sondern auch auf die Struktur der Prozessdefinitionen, die durch Transformation aus dem SPF-Graphen hervorgehen. Der Ausschluss-Constraint wirkt sich hingegen ausschließlich auf den Aufbau des SPF-Graphen aus; während der Transformationsphase muss er hingegen nicht mehr berücksichtigt werden. Umgekehrt, determiniert der Sequenz-Constraint lediglich die Reihenfolge der Aktivitäten in konkreten Prozessdefinitionen und kann während der Konfigurations- bzw. Rekonfigurationsphase vernachlässigt werden. Die hier vorgestellten Constraints repräsentieren also im Hinblick auf ihre Auswirkungen während Konfiguration, Transformation und Rekonfiguration die drei möglichen, unterschiedlichen Klassen. Da SPF-Typ-Graphen prinzipiell keine Einschränkung bezüglich der Constrainttypen definieren, bieten sich die Constraints Anforderung, Ausschluss und Sequenz als Prototypen für alle weiteren, noch zu spezifizierenden Constraints in optimaler Weise an. In Kapitel 4.3.6 wird außerdem thematisiert, dass nicht alle Kombinationen aus logischen Operatoren und Constraints gültig sind. Um nun jedoch zu demonstrieren, in welcher Weise die mögliche Prozessvielfalt durch die eingeführten Konzepte abnimmt, greift die nachfolgende Graphik das Beispiel aus Abbildung 34 auf und erweitert es um die Angabe logischer Verknüpfungen und um den Sequenz-Constraint »Radiologische Untersuchung *folgt auf* Laboruntersuchung«; alle durchgestrichenen Prozessdefinitionen verstoßen nun gegen die getroffenen Festlegungen.

Abbildung 40 Reduktion der Vielfalt möglicher Prozessdefinitionen durch semantische Anreicherung von SPF-Typ-Graphen



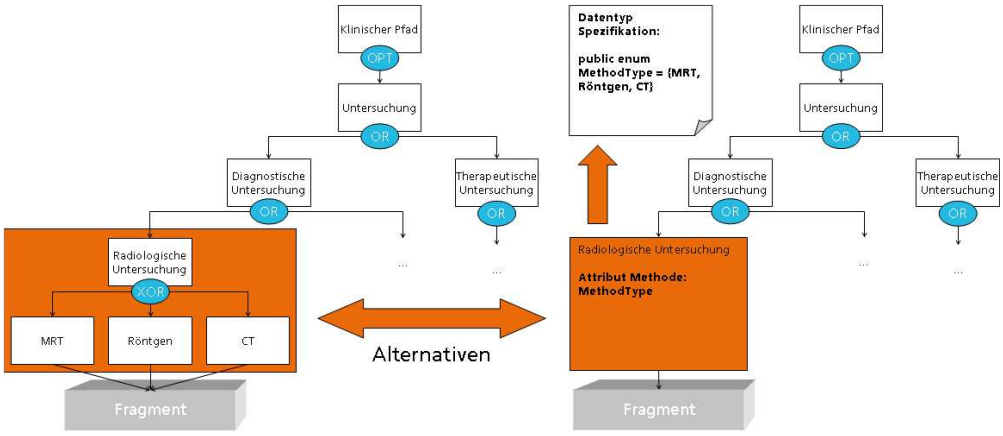
Während also die Prozessmodellierung auf Basis von SPF-Typ-Graphen ohne die Spezifikation logischer Verknüpfungen, Kardinalitäten und Constraints weitgehend der klassischen, deklarativen Prozessmodellierung entspricht, ergibt sich durch die Nutzung dieser Konzepte ein Übergang hin zu einer zunehmend prozeduralen bzw. imperativen Modellierungsmethodik. Infolge der Anreicherung von SPF-Typ-Graphen mit mehr Semantik, lässt sich also der Anteil der prozeduralen Prozessmodellierung erhöhen und Prozessdefinitionen werden insgesamt einheitlicher und sequentieller. Mit der Anzahl solcher Festlegungen im SPF-Typ-Graphen, die sich generell auf alle Prozessdefinitionen auswirken, reduziert sich automatisch die Menge an Modellierungsvarianten, wodurch auch der Aufwand zur Erstellung neuer Prozessdefinitionen abnimmt. Allerdings muss die Entscheidung zur Deklaration von Operatoren wie »AND« und »XOR« oder Constraints im SPF-Typ-Graphen sorgfältig abgewogen werden, da diese Konstrukte zu Einschränkungen der Flexibilität bei der Prozessmodellierung führen. Um also Festlegungen zu treffen, die nicht allgemein, sondern immer nur dann gelten, wenn bestimmte Rahmenbedingungen erfüllt sind, ist die Spezifikation von Prozessvarianten besser geeignet (siehe Kapitel 4.7).

In Declare wird zusätzlich zwischen optionalen Constraints, also Constraints, die bei Bedarf verletzt werden dürfen, und obligatorischen Constraints unterschieden (siehe Kapitel 3.2.4). Da in dieser Arbeit der Fokus auf der Schaffung einer Grundlage zur Unterstützung von Anwendern bei der Modellierung von Prozessdefinitionen und deren Anpassung liegt, wird diese Unterscheidung nicht aufgegriffen. Dennoch soll an dieser Stelle deutlich gemacht werden, dass die letztendliche Entscheidung, wie der Behandlungsprozess für einen Patienten gestaltet werden soll, ausschließlich bei dem behandelnden Arzt liegt und daher im Prinzip gegen alle Vorgaben verstoßen werden könnte, die auf Modellebene vorgenommen wurden. Dies kann allerdings zu erheblichen Problemen führen, wenn die Prozessdefinition nach der Verletzung der Festlegungen im SPF-Typ-Graphen nicht mehr ausführbar ist, weil z. B. Datenabhängigkeiten nicht berücksichtigt wurden. Auch wenn das Modellierungswerkzeug in der Lage ist, derartige Inkonsistenzen zu erkennen, muss bezweifelt werden, ob der Anwender mit sol-

chen Informationen etwas anfangen kann. Sollen also gemäß der SPOT-Vision Fachanwender tatsächlich in die Situation versetzt werden, ausführbare Prozessdefinitionen selbständig zu entwickeln, führt kein Weg an notwendigen Einschränkungen der Modellierungsfreiheit vorbei.

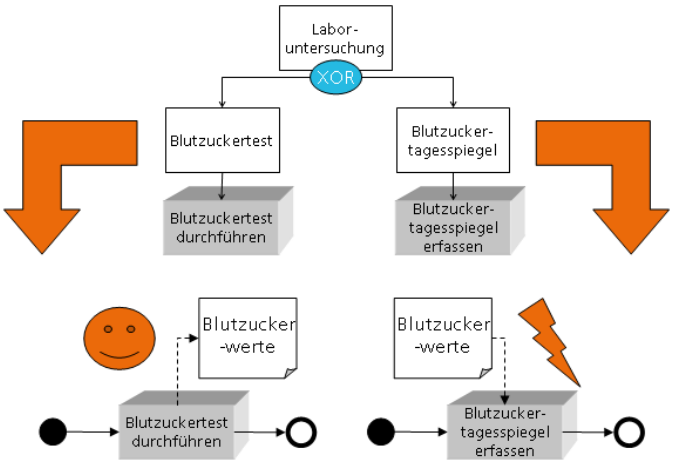
Die letzte Erweiterung, die an SPF-Typ-Graphen vorgenommen werden muss, besteht in der Einführung von Knotenattributen. Im Bereich der Feature-Modellierung wurde ebenfalls die Notwendigkeit eines solchen Konzeptes erkannt [Bednasch 2002, Czarnecki et al. 2002]. Obwohl Attribute auch als Blattknoten eines übergeordneten Elternknoten realisiert werden könnten, wie das z. B. bei attributierten Graphen der Fall ist [Heckel et al. 2002], wird dieses Verfahren hier nicht angewandt. Der Grund hierfür ist der semantische Unterschied, der zwischen einem SPF-Knoten und einem Attribut existiert: Im Gegensatz zu den SPF-Knoten innerhalb des SPF-Typ-Graphen gibt es für Attribute keine separate Realisierung in Form eines Prozessfragments. Stattdessen sind für Attribute Wertzuweisungen möglich, die einem fixen, im SPF-Typ-Graphen festgelegten Datentyp entsprechen müssen und die entweder konstant vordefiniert oder variabel bei der Konfiguration oder auch Rekonfiguration des SPF-Typ-Graphen eingestellt werden können. Unter bestimmten Bedingungen ist es sogar sinnvoll statt SPF-Knoten Attribute zu spezifizieren, um die verschiedenen, möglichen Ausprägungsformen eines Knotens zu reflektieren; in diesem Fall verlagert sich die Spezifikation des SPFs weiter nach oben in der Hierarchie. Die folgende Abbildung illustriert die während der Definitionsphase notwendige Design-Entscheidung, ob die Ausprägungen einer radiologischen Untersuchung als SPF-Knoten oder als Attribut mit dem speziell definierten Datentyp »MethodType« umgesetzt werden soll. Diese Entscheidung hängt im Wesentlichen davon ab, ob es für die alternativen Bild-gebenden Methoden eigene Prozessfragmente geben soll oder nicht und ob zwischen den einzelnen Untersuchungen spezielle Beziehungen in Form logischer Operatoren und Constraints ausgedrückt werden müssen. Gibt es keine solchen Abhängigkeiten und können alle Untersuchungen mit Hilfe desselben Prozessfragments adäquat unterstützt werden, müssen die einzelnen Verfahren nicht als SPF-Knoten modelliert werden; aus Gründen der Komplexitätsreduktion spricht in solchen Fällen hingegen alles für die Nutzung eines entsprechenden Attributs. Das SPF wird dann durch die Zuordnung des Knotens »Radiologische Untersuchung« zu dem entsprechenden, einheitlich zu verwendenden Prozessfragment gebildet. Sollen jedoch Beziehungen zwischen den Prozessfragmenten ausgedrückt werden und sind gegebenenfalls unterschiedliche Prozessfragmente erforderlich, so ist dem Anlegen separater SPF-Knoten der Vorzug zu geben.

Abbildung 41 Gegenüberstellung von Knoten- und Attributdeklarationen in SPF-Typ-Graphen



Mit der Festlegung von Knotenattributen dienen SPF-Typ-Graphen nun auch der hierarchischen Organisation eines Parameterraumes [Czarnecki et al. 2002]. In Anbetracht der guten Erfahrungen mit Feature-Modellen in einer Vielzahl von Anwendungsbereichen, sind SPF-Typ-Graphen nach der Einführung der hier beschriebenen Konzepte ausdrucksmächtig genug, um das Prozesswissen innerhalb einer Domäne im Hinblick auf den funktionalen und verhaltensorientierten Workflow Aspekt adäquat beschreiben zu können. Da SPF-Typ-Graphen nicht dazu dienen, sämtliche Workflow Aspekte in einem einzigen Modell zusammenzufassen, werden Informationen zu Datenobjekten oder Ressourcen nicht in den SPF-Typ-Graphen integriert. Welche Datenobjekte und Ressourcen für die Ausführung eines Prozessfragments benötigt werden, ist vielmehr in der Schnittstellenspezifikation des entsprechenden Fragments hinterlegt. Dieser Umstand kann bei der Definition des SPT Typ Graphen jedoch zu Problemen bei der Konsistenz des Datenflusses in resultierenden Prozessdefinitionen führen. Die nachfolgende Abbildung demonstriert einen solchen Fall. Mit dem XOR-Operator wird festgelegt, dass ein Blutzuckertest und ein kompletter Blutzuckertagesspiegel nur alternativ zueinander durchgeführt werden können. Andererseits ist in der Schnittstellenbeschreibung definiert, dass für den Blutzuckertagesspiegel ein Datenobjekt benötigt wird, das zuvor nur durch den Blutzuckertest produziert werden kann. Diese Spezifikationen führen also gezwungenermaßen zu einer inkorrekten Prozessdefinition, wenn das vom Prozessfragment »Blutzuckertagesspiegel« benötigte Datenobjekt nicht durch ein anderes Prozessfragment bereitgestellt wird.

Abbildung 42 Problematik inkonsistenter Datenflüsse durch Separation der Workflow Aspekte



Die Festlegung logischer Operatoren und Constraints im SPF-Typ-Graphen muss sich also an den Schnittstellenspezifikationen der zugeordneten Prozessfragmente orientieren. Die Entwicklung automatischer Prüfroutinen, die die Konsistenz der Struktur des SPF-Typ-Graphen mit den Anforderungen aus den Prozessfragmenten sicherstellt, ist nicht Teil dieser Arbeit. In Kapitel 4.5.4 wird jedoch am Beispiel einer konkreten Prozessmodellierungssprache gezeigt, welche Inkonsistenzen sich prinzipiell ergeben können und wie diese durch adäquate Gestaltung des SPF-Typ-Graphen vermieden werden können.

4.3.4 Formale Spezifikation von Prozessfragmenten

Mit der Einführung des Konzepts der Prozessfragmente soll die Entwicklung wieder verwendbarer Bausteine und ihr Einsatz in unterschiedlichen Prozessdefinitionen für beliebige WfMS

gefördert werden. Während im SPOT-Projekt keine exakten Festlegungen im Hinblick auf die Schnittstellenspezifikation von Prozessfragmenten gemacht wurden, wird in diesem Kapitel eine Definition gegeben, bei der auch die Vorgaben für ADEPT2 Aktivitätsvorlagen berücksichtigt wurden [Reichert 2000b]. Dies bietet sich zum einen aufgrund der existierenden Parallelen bei den beiden Konzepten an (siehe Diskussion in Kapitel 4.1.2) und zum anderen, da der Transformationsprozess eines getypten SPF-Graphen in eine Prozessdefinition am Beispiel des WfMS ADEPT2 demonstriert wird (siehe Kapitel 4.5).

Sei *ID* die Menge aller atomaren, eindeutigen Identifikationen und $ProcessfragmentIds \subseteq ID$ die Menge aller möglichen Identifikationen für Prozessfragmente; dann entspricht das folgende Tupel den allgemeinen Attributen eines Prozessfragments mit der Identifikation $p \in ProcessfragmentIds$:

$$(Name^p, Desciptionr^p, Version^p, Role^p, Method^p, Implementation^p)^{35}$$

Mit diesem Tupel werden die grundlegenden Attribute zur Beschreibung der Schnittstelle eines Prozessfragments zusammengefasst. Das Attribut $Name^p$ entspricht einer Bezeichnung für das Fragment und der Benennung der Aktivität, die in die entsprechende Prozessdefinition eingebettet wird. Auf diese Weise soll das Verständnis der Bedeutung des Fragments und ihr Auffinden für menschliche Anwender erleichtert werden. Mit $Description^p$ kann eine detaillierte Beschreibung der mit diesem Fragment erreichten Funktionalität gegeben werden. $Version^p$ repräsentiert die jeweilige Versionsnummer des Prozessfragments; neue Versionen können automatisch erstellt werden, sobald sich Änderungen an den Attributen der Schnittstellenspezifikation ergeben. Das Attribut $Role^p$ ermöglicht die Zuordnung einer Rollenspezifikation aus einem Organigramm; dies ist notwendig, wenn das Prozessfragment eine manuell durchgeführte Tätigkeit repräsentiert, die lediglich IT-gestützt ablaufen soll (z. B. durch Erfassung von Daten in einer entsprechenden Eingabemaske). Da Anwendungskomponenten meist nicht nur der Durchführung einer einzigen Funktion dienen, sondern mehrere Funktionen zusammenfassen, besteht mit der Angabe des Attributs $Method^p$ die Möglichkeit, eine bestimmte Methode der Anwendungskomponente auszuwählen, die aufgerufen werden soll; dabei kann es sich z. B. um die einzelne Operation eines Web Services oder die Methode einer Java Klasse handeln. Das Attribut $Implementation^p$ verweist auf die tatsächliche Implementierung der Anwendungskomponente, wie z. B. durch Angabe der entsprechenden WSDL (Web Service Description Language) oder des Speicherortes der benötigten Jar-Datei (Java archive). Die nachfolgende Tabelle listet zwei Beispiele für mögliche Attributbelegungen bei der Schnittstellenspezifikation von Prozessfragmenten auf.

[35] Für eine Abbildungsfunktion f: X ↦ Y werden die Werte f(x) in der verkürzten Schreibweise $f^x \in$ Y dargestellt.

Abbildung 43 Beispiele für Schnittstellenspezifikationen von Prozessfragmenten

Identifikation Attribut	**1.0.23.7.879.101**	**1.0.23.7.879.158**
Name^{p}	Radiologische Untersuchung durchführen	Entlassbrief verfassen
Description^{p}	Durchführung einer Bild-gebenden Methode im Rahmen der Diagnostik	Verfassen eines Arztbriefes an den einweisenden und/oder weiterbehandelnden Arzt
Version^{p}	2	1
Role^{p}	Radiologe/-in	Stationsarzt/-ärztin
Method^{p}	1.0.23.7.447.329	1.0.23.7.447.702
Implementation^{p}	1.0.23.9.53.818	1.0.23.8.106.872

Der Grund für Abweichungen zwischen der Schnittstellenspezifikation für Aktivitätsvorlagen in ADEPT2 und den hier verwendeten Prozessfragmenten besteht vor allem darin, dass viele der ADEPT2 Attribute speziell für das eigene WfMS entwickelt wurden, während die Schnittstellenspezifikationen von Fragmenten zunächst unabhängig von spezifischen WfMS sein müssen. Soll ein Prozessfragment z. B. das Starten eines Subprozesses bewirken, ist dies bei der Schnittstellenbeschreibung des Fragments nicht von Bedeutung, da die Ausführung des Subprozesses grundsätzlich außerhalb der Kontrolle des WfMS liegen kann, das für die Durchführung der eigentlichen Prozessdefinition zuständig ist, die aus dem SPF-Typ-Graphen abgeleitet wurde. In ADEPT2 wird hingegen mit Hilfe eines speziellen Attributs angezeigt, ob es sich um einen Subprozess handelt, da diese Art von Implementierung dann natürlich von ADEPT2 selbst behandelt wird. Darüber hinaus ist der Ort, an dem die Implementierung liegt, bei ADEPT2 statisch, weil es sich immer um das lokale WfMS-spezifische Repository handelt.

Da Methoden über eigene Attribute verfügen, liefert $Method^{p}$ lediglich die Referenz auf die konkrete Spezifikation der Methode. $MethodIds \subseteq ID$ ist die Menge aller atomaren Identifikationen für Methoden. Jedem Prozessfragment $p \in ProcessfragmentIds$ ist genau eine Methode $Method^{p} \subseteq MethodIds$ zugeordnet, wobei die Schnittmenge der Identifikationen für Prozessfragmente und Methoden leer ist:

$$MethodIds \cap ProzessfragmentIds = \varnothing$$

Demzufolge können auch unterschiedliche Prozessfragmente dieselbe Methode anbieten. Jedes Element $m \in Method^{p}$ wird durch folgendes Tupel repräsentiert:

$$(Name^{m}, Description^{m}, Parameters^{m})$$

Die nachfolgende Tabelle verdeutlicht beispielhaft mögliche Attributwerte für Methoden.

Tabelle 17 Beispiele für Attributwerte von Methoden

Identifikation Attribut	**1.0.23.7.447.329**	**1.0.23.7.447.702**
Name^{m}	makeNewImage	loadTemplate
Description^{m}	Neues radiologisches Bild in der elektronischen Patientenakte speichern.	Personalisierte Vorlage für Arztbriefe laden.
Parameters^{m}	1.0.23.7.775.123, 1.0.23.7.775.122, 1.0.23.7.775.156	1.0.23.7.775.123, 1.0.23.7.775.122, 1.0.23.7.88.991

Die Datenobjekte, die eine Methode als Eingabeparameter erhält und als Ausgabeparameter zurückgibt, werden über das Attribut $Parameters^{m}$ referenziert. $ParameterIds \subseteq ID$ entspricht der Menge aller atomaren Identifikationen für Parameter. Jeder Methode mit Identifikation

$m \in MethodIds$ ist eine Menge von Parametern $Parameters^m \subseteq ParametersIds$ zugeordnet, wobei gilt, dass die Schnittmenge sämtlicher Identifikationsmengen der leeren Menge entspricht:

$$ParameterIds \cap MethodIds \cap ProzessfragmentIds = \varnothing$$

Das folgende Tupel legt fest, welche Attribute für jeden Parameter $param \in Parameters^m$ spezifiziert werden:

$$(Name^{param}, Description^{param}, IOtype^{param}, Demandmode^{param}, Datatype^{param})$$

Wie für Prozessfragmente so kann auch für Parameter durch die Attribute $Name^{param}$ und $Description^{param}$ eine Bezeichnung und eine Beschreibung angegeben werden. Das Attribut $IOtype^{param} \in \{IN, OUT, IN/OUT\}$ legt fest, ob ein Parameter im Kontext eines Prozessfragments als Eingabe- oder Ausgabeobjekt dient; dementsprechend erhält das Attribut entweder den Wert »IN«, »OUT« oder »IN/OUT«. Der letztgenannte Wert wird z. B. verwendet, wenn das Datenobjekt durch die Anwendungskomponente manipuliert und anschließend in den Datenkontext des WfMS zurückgeschrieben werden soll. Mit $Demandmode^{param} \in$ {MANDATORY, OPTIONAL} wird festgelegt, ob ein Parameter unbedingt notwendig ist, damit die Anwendungskomponente korrekt ausgeführt werden kann oder ob der Parameter nur optional vorliegen muss. Mit $Datatype^{param}$ wird der Wertebereich des Parameters bestimmt. Dabei kann es sich um die in einem Rechner direkt darstellbaren Basistypen, wie z. B. Zeichenketten, Bool'sche Werte oder ganze Zahlen, handeln; ebenso müssen auch komplexe Datentypen, die z. B. im HL7 Information Reference Modell (HL7 RIM)[36] definiert wurden, oder benutzerspezifische Datentypen möglich sein. Das hier beschriebene Konzept macht es aber generell einfacher, einem WfMS komplexe Datentypen bekannt zu machen, als dies bei traditionellen Systemen der Fall ist. Der Grund dafür ist, dass WfMS nur noch die Anforderung erfüllen müssen, beliebige Datenobjekte in ihrem Datenkontext speichern zu können und kontrolliert mit den externen Anwendungskomponenten auszutauschen; es ist aber nicht mehr notwendig, dass WfMS in der Lage sind, die Objekte in ihrem Datenkontext selbständig zu interpretieren und zu verarbeiten, um daraus die korrekte Steuerung des Prozesses ableiten zu können. Derzeitige WfMS führen eine ganze Reihe von operativen Aktionen selbständig aus; dazu gehört z. B. die Auswertung Bool'scher Ausdrücke im Kontext bedingter Verzweigungen. Wie wir bei der Transformation von SPF-Graphen zu ausführbaren Prozessdefinitionen jedoch sehen werden, sind solche Vorgänge im Rahmen der hier spezifizierten Methodik nicht mehr notwendig (siehe Kapitel 4.6). Aus diesem Grund können WfMS alle Datenobjekte generell wie eine »Black Box« behandeln, deren interne Struktur und Semantik sie nicht verstehen müssen, sondern die zu Zwecken der Interpretation, Verarbeitung, Darstellung und Manipulation an entsprechende Anwendungskomponenten übergeben werden. Welche Datentypen innerhalb der Schnittstellenspezifikation eines Prozessfragments festgelegt werden, ist daher nur für die Implementierung des Fragments durch die jeweilige Anwendungskomponente und für die Prüfung der Korrektheit des Datenflusses durch das WfMS von Bedeutung. Die nachstehende Tabelle zeigt Beispiele für mögliche Wertebelegungen der Parameterattribute.

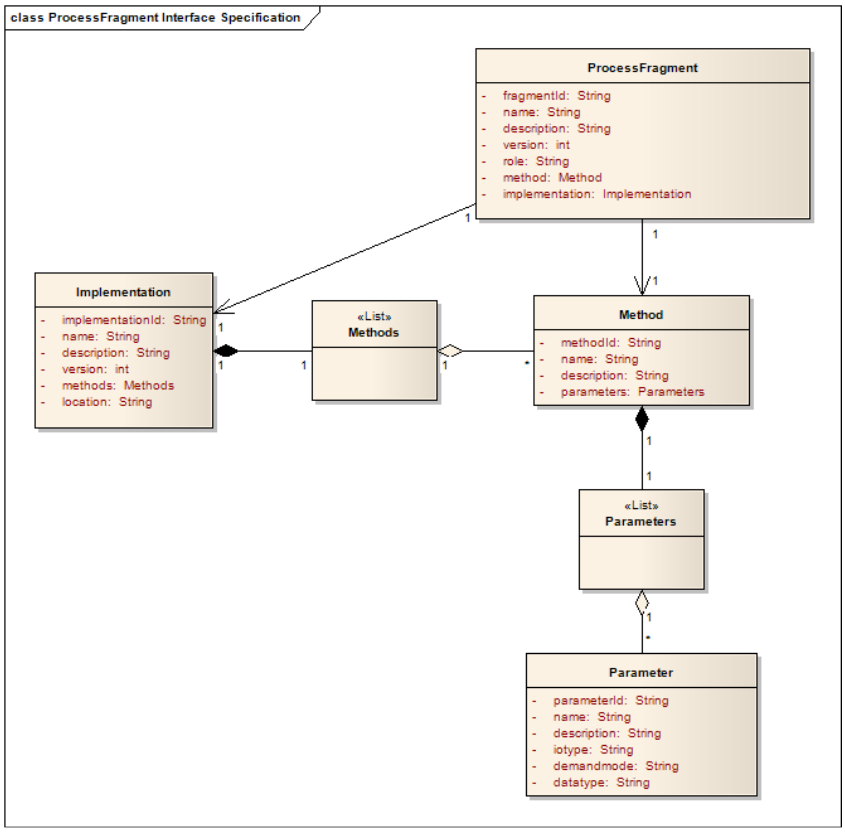
[36] Siehe http://www.hl7.org/Library/data-model/RIM/modelpage_mem.htm für die aktuell gültige Version des HL7 RIM

Abbildung 44 Beispiele für Attributwerte von Parametern

Identifikation Attribut	**1.0.23.7.775.123**	**1.0.23.7.775.156**
Name[param]	PatientId	Method
Description[param]	Eindeutige Identifikation eines Patienten	Radiologische Untersuchungsmethode
IOtype[param]	IN	IN
Demandmode[param]	MANDATORY	MANDATORY
Datatype[param]	HL7:II[37]	Userdefined: MethodType

Die folgende Abbildung illustriert die Schnittstellenspezifikation für Prozessfragmente als UML Klassendiagramm.

Abbildung 45 Schnittstellenspezifikation für Prozessfragmente als UML Klassendiagramm



Damit ein Prozessfragment vollständig ist und zusammen mit dem SPF-Knoten eines SPF-Typ-Graphen ein ausführbares semantisches Prozessfragment bilden kann, muss ihm eine Implementierung zugeordnet werden, die im Attribut *Implementation*^{p} referenziert wird. Die Implementierung entspricht einem Anwendungsprogramm, das nach der Instanziierung des

[37] HL7:II entspricht dem als Basisdatentyp in HL7 spezifizierten komplexen Identifizierer; mögliche Bestandteile sind z.B. OID (Object Identifier) oder UUID (Universally Unique Identifier)

entsprechenden Prozessfragments vom WfMS mit den Eingabeparametern aus dem Datenkontext der Prozessinstanz aufgerufen wird. Die tatsächliche Ausführung des Anwendungsprogramms läuft außerhalb der Kontrolle des WfMS ab. Grundsätzlich muss die Verknüpfung zwischen Schnittstellenspezifikation des Prozessfragments und seiner Implementierung nicht statisch sein. Wie wir in Kapitel 3.2.2 gesehen haben, kann die Zuordnung auch dynamisch z. B. durch die Auswertung eines Regelbaumes nach der RDR Methodik erfolgen. Da im Krankenhaus die verfügbaren bzw. benötigten Anwendungsprogramme jedoch im Voraus klar definiert werden können und in der Regel keine dynamische Auswahl aus einer Menge alternativer Implementierungsarten möglich ist, werden Methoden zum dynamischen Binden von Anwendungskomponenten in dieser Arbeit nicht weiter vertieft. Stattdessen kann grundsätzlich davon ausgegangen werden, dass die Verknüpfung von Schnittstellenbeschreibung und Implementierung sowie die Zuordnung zu einem SPF-Knoten des SPF-Typ-Graphen während der Definitionsphase erfolgen und zur Laufzeit des Prozesses statisch sind. Die Voraussetzung für das Verbinden einer Schnittstellenspezifikation und einer Implementierung zu einem vollständigen Prozessfragment ist die so genannte »Plug-in-Kompatibilität« der beiden Komponenten [Reichert 2000b]. Die Plug-in-Kompatibilität definiert, welche Anforderungen eine Implementierung mindestens erfüllen muss, um der Schnittstellenbeschreibung eines Fragments zu entsprechen. Das bedeutet zum einen, dass die Anwendungskomponente die in der Schnittstellenspezifikation angegebene Methode bereitstellt, und dass zum anderen alle obligaten Eingabeparameter der Anwendungskomponente gemäß der Schnittstellenspezifikation des Prozessfragments korrekt versorgt werden und umgekehrt, die Implementierung die von der Spezifikation geforderten Ausgabeparameter zurückliefern kann. Analog zu Reichert lässt sich dieser Sachverhalt auf einen Abgleich der Signaturen der Schnittstellenspezifikation und der Implementierung zurückführen. Die für ADEPT2 spezifizierten Bedingungen für Plug-in-Kompatibilität müssen jedoch so geändert werden, dass sie der Schnittstellenbeschreibung für Prozessfragmente gerecht werden. Dementsprechend ist eine Implementierung *c* plug-in-kompatibel zur Schnittstellenspezifikation eines Prozessfragments *p (*Schreibweise: c $\vDash$ p), wenn folgende Faktoren erfüllt sind:

> *Sei pm = Method^{p} die in der Schnittstellenspezifikation von p referenzierte Methode; dann folgt:* $\forall$ *c* $\in$ *Implementation^{p}* $\exists$ *cm* $\in$ *Methods^{c}: pm = cm*
>
> $\forall$ *x* $\in$ *Parameters^{cm} mit (IOtype^{x} = IN* $\vee$ *IOtype^{x} = IN/OUT)* $\wedge$ *Demandmode^{x} = MANDATORY* $\Rightarrow$ $\exists$ *y* $\in$ *Parameters^{pm}: ((IOtype^{y} = IN* $\vee$ *IOtype^{y} = IN/OUT)* $\wedge$ *Demandmode^{y} = MANDATORY)* $\wedge$ *Datatype^{y}* $\subseteq$ *Datatype^{x}*
>
> $\forall$*u* $\in$ *Parameters^{pm} mit (IOtype^{u} = OUT* $\vee$ *IOtype^{u} = IN/OUT)* $\wedge$ *Demandmode^{u} = MANDATORY* $\Rightarrow$ $\exists$ *v* $\in$ *Parameters^{cm}: ((IOtype^{v} = OUT* $\vee$ *IOtype^{v} = IN/OUT)* $\wedge$ *Demandmode^{v} = MANDATORY)* $\wedge$ *Datatype^{v}* $\subseteq$ *Datatype^{u}*

Diesen Definitionen zufolge sorgt die Plug-in-Kompatibilität dafür, dass die Anwendungskomponente *c* über die in *p* benannte Methode verfügen muss und maximal dieselben obligaten Eingabeparameter fordern darf, die in der Schnittstellenspezifikation der Methode *pm* des Prozessfragments *p* als verpflichtend festgelegt sind; analog dazu, darf *pm* nicht mehr obligate Ausgabeparameter benötigen, als *cm* sicher zurückliefert. Um die Plug-in-Kompatibilität zu gewährleisten, kann es notwendig sein, entsprechende Datentypkonversionen durchzuführen; diese werden im Rahmen dieser Arbeit jedoch nicht betrachtet. Wird jede Implementierung an Hand von den in Abbildung 45 festgelegten Metadaten beschrieben, ist ein Abgleich zwischen der Schnittstellenspezifikation des Prozessfragments und der der Implementierung einfach zu realisieren. Ob die Metadaten aus der Schnittstellenbeschreibung der Implementierung jedoch

auch tatsächlich den dort umgesetzten Methoden entsprechen, ist hingegen nur durch eine aufwändige Code-Analyse feststellbar. Alternativ zur komplett automatischen Überprüfung der Plug-in-Kompatibilität kann diese auch manuell durch den Entwickler, der die Verknüpfung zwischen Fragment und Implementierung vornimmt, sichergestellt werden.

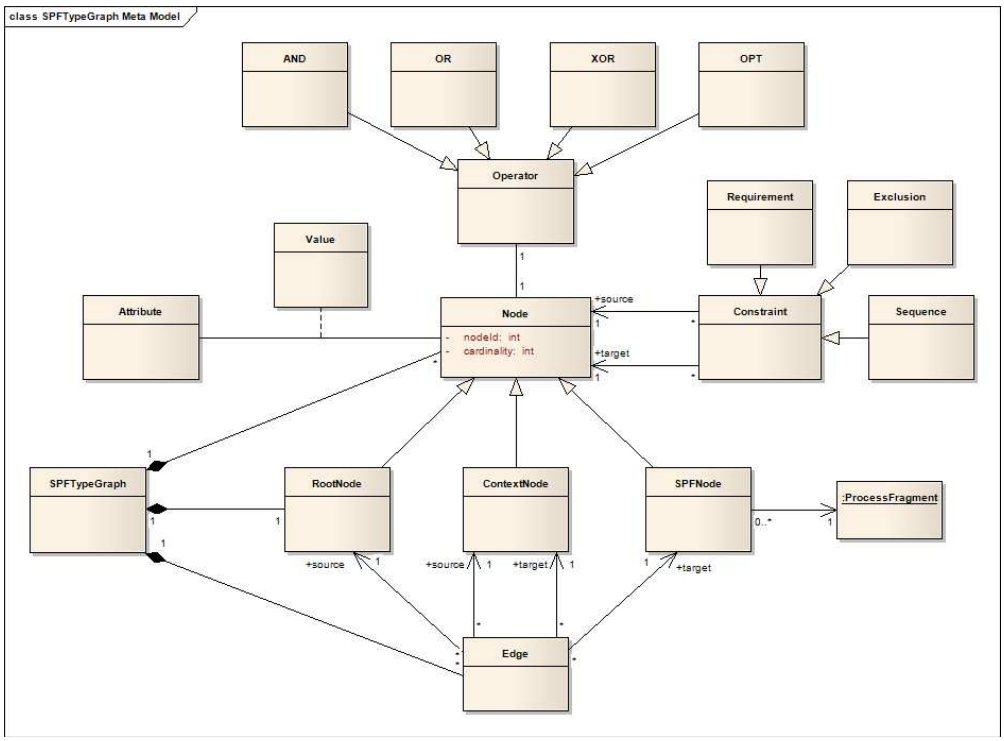
Durch die Aufteilung von Prozessfragmenten in eine Spezifikations- und eine Implementierungskomponente wird einerseits erreicht, dass bereits entwickelte Anwendungsprogramme durch die nachträgliche Zuordnung einer Schnittstellenbeschreibung in eine Prozessdefinition eingebettet werden können; andererseits kann die Schaffung neuer Anwendungskomponenten durch die klaren Vorgaben bei der vorausgehenden Spezifikation der Schnittstellenbeschreibung gesteuert werden. Nicht zuletzt ist das Vorliegen einer Schnittstellenbeschreibung die Voraussetzung für das sinnvolle Kombinieren von Prozessfragmenten zu kompletten Prozessdefinitionen. Im Kontext der Prozessdefinition kann nun nämlich geprüft werden, ob die obligaten Eingabeparameter von Prozessfragmenten vorher von anderen Fragmenten als Ausgabeparameter bereitgestellt werden; dies ist die grundlegende Bedingung für die Sicherstellung der Korrektheit des Datenflusses in Prozessdefinitionen [Reichert 2000b].

Obwohl im Hinblick auf die Implementierung keine Vorgaben gemacht werden, wird hier der Einsatz plattformunabhängiger Technologien empfohlen. Für die Durchführung singulärer Aktivitäten können z. B. Web Services genutzt werden. Bei der Spezifikation von Prozessfragmenten als eigene Prozessdefinitionen, die als Subprozesse in übergeordnete Definitionen integriert werden sollen, bieten sich Modellierungssprachen an, die den Einsatz unterschiedlicher WfMS ermöglichen. Im SPOT-Projekt wurde zu diesem Zweck das Metamodell SPOT-MM entwickelt, mit dem Prozessdefinitionen erstellt und durch den SPOT-Process-Compiler, auf unterschiedlichen WfMS eingerichtet und ausgeführt werden können.

4.3.5 Formale Spezifikation von SPF-Typ-Graphen

Nach der informalen Definition von SPF-Typ-Graphen und deren Eigenschaften in Kapitel 4.3.3 folgt nun die formale Spezifikation des Konzeptes [Reuter 2010]. Nur auf Grundlage einer eindeutigen formalen Semantik kann überprüft werden, ob die Instanz eines Konzeptes dessen Anforderungen genügt und damit als »korrekt« bezeichnet werden kann. Darüber hinaus ist die Formalisierung von Bedeutung, um Vorgänge in Zusammenhang mit dem Konzept stärker zu automatisieren; im Hinblick auf SPF-Typ-Graphen ist neben der Möglichkeit zur Durchführung von Korrektheitsprüfungen vor allem die Unterstützung bei der Konfiguration durch den Fachanwender und die automatisierte Ableitung von Prozessdefinitionen zu nennen. Um einen Überblick über den Aufbau von SPF-Typ-Graphen zu erhalten, illustriert Abbildung 46 zunächst deren grundlegende Struktur in einem UML Klassendiagramm.

Abbildung 46 Metamodell für SPF-Typ-Graphen als UML Klassendiagramm



Aus dem UML Klassendiagramm lassen sich bereits einige generelle Eigenschaften von SPF-Typ-Graphen ablesen, die erfüllt sein müssen, damit es sich um eine gemäß Meta Modell korrekte Instanz handelt. So wird durch die Aggregationsbeziehung zwischen den Klassen »SPFTypeGraph« und »RootNode« festgelegt, dass ein vollständiger und korrekter SPF-Typ-Graph genau einen Wurzelknoten umfassen muss. Gemäß der Beziehung zwischen »SPF-TypeGraph« und den Klassen »ContextNode« und »SPFNode« kann jeder Graph beliebig viele Kontext- und SPF-Knoten enthalten; wie sich an den »source« und »target« Relationen erkennen lässt, kann ein SPF-Knoten im Gegensatz zur Wurzel und den Kontextknoten nicht Ausgangspunkt, sondern nur Ziel einer gerichteten Kante bzw. einer hierarchischen Relation sein kann. Das UML Diagramm macht jedoch keinerlei Aussagen darüber, ob tatsächlich alle Knoten innerhalb des SPF-Typ-Graphen über einen Pfad aus hierarchischen Relationen mit der Wurzel verbunden sein müssen oder ob es innerhalb des Graphen Zyklen geben darf.

Wie bereits erwähnt, wurde in den letzten Jahren verstärkt der Bedarf nach der Formalisierung von Feature-Modellen erhoben. Da SPF-Typ-Graphen spezialisierte Feature-Modelle zur Abbildung des Prozesswissens einer Domäne repräsentieren, berücksichtigt das in dieser Dissertation aufgestellte formale Modell daher vorausgegangene Formalisierungsarbeiten im Umfeld der Feature-Modellierung [Mannion 2002, Czarnecki et al. 2005, Sun et al. 2006, Schobbens et al. 2006]. Während die meisten Formalisierungsansätze sich der »First Order Logic« bedienen, um Korrektheitskriterien für Feature-Modelle und deren Konfigurationen aufzustellen, beschreibt Czarnecki et al. das Feature-Modell in Form einer Graphgrammatik. Dieser Ansatz hat den Vorteil, dass auch dem Konfigurationsvorgang ein formales Modell zugrunde gelegt werden kann. Werden lediglich Korrektheitskriterien für Feature-Modelle und deren Konfiguration aufgestellt, so kann immer nur im Nachhinein überprüft werden, ob eine Konfiguration den Regeln des Feature-Modells genügt. Indem mit einer Graphgrammatik die Menge an Produktionsregeln festgelegt wird, die zur Konfiguration des Feature-Modells verwendet

werden, sind inkorrekte Konfigurationen dagegen von Vorneherein ausgeschlossen. Dieses Vorgehen korreliert mit dem Ansatz der Prozessmodellierung auf Basis komplexer Änderungsoperationen, wie er von ADEPT2 verfolgt wird (siehe Kapitel 3.2.6). Wenn Fachanwender tatsächlich in der Lage sein sollen, Prozessdefinitionen selbst zu erstellen und bei Bedarf zu modifizieren, ist diese Formalisierung der Konfiguration also von großer Bedeutung. Darf ein Anwender nämlich ausschließlich die Produktionsregeln ausführen, die ihm von der Graphgrammatik angeboten werden, kann sichergestellt werden, dass er nur solche Konfigurationen kreiert, die tatsächlich in gültige Prozessdefinitionen überführt werden können.

Die Grundlage für die Graphgrammatik und korrekte Durchführung der Konfiguration bildet jedoch nach wie vor eine gültige Instanz des Meta Modells für SPF-Typ-Graphen. Aus diesem Grund wird in diesem Kapitel zunächst eine formale Definition für SPF-Typ-Graphen gegeben. Darüber hinaus werden Kriterien aufgestellt, an Hand derer entschieden werden kann, ob es sich um einen korrekten Graph handelt. Auf die Ableitung einer Graphgrammatik aus SPF-Typ-Graphen zur Konfigurationsunterstützung wird hingegen erst in Kapitel 4.4 detailliert eingegangen.

Definition (SPF-Typ-Graph). Ein SPF-Typ-Graph ist ein Tupel $STG = (V, E, V_{SPF}, r, E, C, A, VO, VC, VP, VA)$, wobei die Elemente folgendermaßen definiert sind:
- V (endliche, nicht leere Knotenmenge)
- $E \subseteq V \times V$ (endliche Menge von hierarchischen Knotenrelationen)
- $V_{SPF} \subseteq V$ (endliche, nicht leere Menge von SPF-Knoten)
- $r \in V$ (Wurzelknoten)
- $C \subseteq V \times V \times ConstraintType$ (endliche Menge von Constraint-Relationen)
- A (endliche Menge von Knotenattributen)
- $VO: V \setminus V_{SPF} \nrightarrow OperatorType$ (Operatorenfunktion)
- $VC: V \nrightarrow \mathbb{N} \setminus 0$ (Kardinalitätsfunktion)
- $VP: V_{SPF} \nrightarrow P$ (Fragmentzuordnungsfunktion)
- $VA: V \times A \times AttributeType \nrightarrow \{Undefined\}$ (Attributwertfunktion)

Den einzelnen Komponenten des SPF-Typ-Graphen kommt dabei die folgende Bedeutung zu:
- V ist die endliche, nicht leere Menge aller Knoten bzw. eindeutigen Knotenbezeichnungen innerhalb von SPF-Typ-Graphen einschließlich Wurzel, Kontextknoten und SPF-Knoten.
- $E \subseteq V \times V$ entspricht der endlichen Menge aller hierarchischen Relationen über die zwei Knoten miteinander verbunden werden. Seien $v \in V$ und $u \in V$ zwei Knoten innerhalb des SPF-Typ-Graphen; dann bedeutet der Ausdruck $(v, u) \in E$, dass zwischen den Knoten v und u eine gerichtete hierarchische Kante existiert. Diese Kante impliziert, dass v der übergeordnete Knoten von u ist. Zusätzlich definieren wir an dieser Stelle die beiden folgenden Projektionsfunktionen auf die Menge E:
  $source: E \nrightarrow V$ (die »Quellfunktion« liefert den »Quellknoten«, von dem die Kante ausgeht)
  $target: E \nrightarrow V$ (die »Zielfunktion« liefert den »Zielknoten«, zu dem die Kante hinführt)
- $V_{SPF} \subseteq V$ ist die Menge aller SPF-Knoten des Graphen. Die Menge lässt sich folgendermaßen definieren:
  $V_{SPF} = \{v \in V \mid \nexists\, u \in V: (v, u) \in E\}$
- $r \in V$ repräsentiert den Wurzelknoten des SPF-Typ-Graphen, für den gilt:
  $\nexists\ v \in V: (v, r) \in E$
- $C \subseteq V \times V \times ConstraintType$ ist die Menge der Constraint-Relationen, über die zwei Knoten auf horizontaler Achse miteinander in Beziehung gesetzt werden können. Welche Typen für C existieren, wird durch die Menge $ConstraintType = \{REQUIREMENT, SEQUENCE,$

EXCLUSION} festgelegt. Seien $v \in V$ und $u \in V$ zwei Knoten innerhalb des SPF-Typ-Graphen; dann bedeutet der Ausdruck $(v, u, SEQUENCE) \in C$, dass zwischen den Knoten *v* und *u* ein Sequenz-Constraint von *v* nach *u* existiert.

- *VO:* $V \rightarrow$ *OperatorType* bildet den Wurzelknoten und jeden Kontextknoten auf einen logischen Operator ab, über den die Beziehung zu den nachfolgenden Knoten dargestellt wird. Die Menge *OperatorType* ist folgendermaßen definiert (siehe auch Kapitel 4.3.3):
  OperatorType = {AND, OR, XOR, OPT}
- *VC:* $V \rightarrow \mathbb{N} \setminus 0$ ordnet jedem Knoten eine Kardinalität zu, die der maximalen Anzahl möglicher Klone entspricht, die für diesen Knoten gebildet werden dürfen. Im SPF-Typ-Graphen wird für die Kardinalität kein Intervall angegeben, sondern lediglich die maximale Obergrenze definiert. Wenn ein Knoten vom Fachanwender konfiguriert wird, schließt dies automatisch mit ein, dass der Knoten mindestens einmal in der Konfiguration bzw. dem SPF-Graphen vorkommt; umgekehrt, soll ein Knoten nicht im SPF-Graphen und letztlich in der Prozessdefinition repräsentiert werden, so wird er zuvor auch nicht konfiguriert.
- *VP:* $V_{SPF} \rightarrow$ *P entspricht* einer Abbildungsfunktion, die jedem SPF-Knoten innerhalb von STG ein Prozessfragment aus der Menge *P* zuweist:
  $VP(V) \subseteq P$
- *A* ist eine endliche Menge von Knotenattributen. Attribute können verwendet werden, um Prozessfragmente prozessspezifisch zu konfigurieren (z. B. Auswahl der radiologischen Untersuchung). Sie können aber auch der Interoperabilität mit anderen Systemen dienen. So ermöglicht die Kombination aus den Attributen »code«, »codeSystem« und »codeSystemVersion« die Zuordnung zu einem Element aus bereits existierenden Codesystemen, wie z.B. HL7 RIM, LOINC (Logical Observation Identifiers Names and Codes)[38] oder Snomed CT (Systemized Nomenclature for Medicine-Clinical Terms)[39]. Dies hat auch den Vorteil, dass nicht alle semantischen Informationen als Knotenattribute im SPF-Typ-Graphen neu definiert werden müssen, sondern vorhandene Quellsysteme genutzt werden können. Zu jedem Knotenattribut muss im SPF-Typ-Graphen der Datentyp spezifiziert werden, den ein möglicher Wert des Attributs annehmen muss; dies geschieht über eine Datentypfunktion *DT*. Sei $DataType \neq \varnothing$ die Menge möglicher Datentypen:
  DT: $A \rightarrow$ *DataType (Datentypfunktion)*
  Darüber hinaus kann mittels Attributtypfunktion *AT* festgelegt werden, ob es sich um einen obligatorisches oder einen optionales Attribut handelt:
  AT: $V \times A \rightarrow$ *{MANDATORY, OPTIONAL} (Attributtypfunktion)*
  Allen obligatorischen Attributen muss vor Start der Transformation in eine ausführbare Prozessdefinition ein Wert zugewiesen worden sein; dies kann entweder statisch auf Ebene des SPF-Typ-Graphs oder dynamisch während der Konfiguration erfolgen. Die Wertzuweisung ist z. B. dann obligatorisch, wenn sie das Verhalten der Implementierung des Prozessfragments beeinflussen soll.
- *VA:* $V \times A \rightarrow$ *{Undefined}* ordnet einem Knotenattribut einen Wert zu, der dem für das Attribut aus der Menge *A* spezifizierten Datentyp entsprechen muss. Die Zuordnung von Knotenattributen kann bereits zur Definitionsphase stattfinden; diese statische Wertzuweisung entspricht dann der Standardeinstellung des Knotens. Fachanwender haben jedoch die

[38] Für mehr Informationen zu LOINC (Logical Observation Identifiers Names and Codes) siehe http://loinc.org/

[39] Für mehr Informationen zu Snomed CT (Systemized Nomenclature for Medicine-Clinical Terms) siehe http://www.ihtsdo.org/snomed-ct/

Möglichkeit, Attribute während der Konfigurations- bzw. Rekonfigurationsphase zu verändern.

Zum Abschluss dieses Kapitels werden zwei verkürzte Schreibweisen eingeführt. Seien $v, u \in V$ zwei Knoten innerhalb eines SPF-Typ-Graphen:
- $v \rightarrow u \Leftrightarrow (v, u) \in E$ *(u ist direkter Nachfolger von v)*
- $v \rightarrow^* u \Leftrightarrow v \rightarrow u \vee (\exists z \in V: v \rightarrow z \wedge z \rightarrow^* u)$ *(es existiert ein Pfad aus hierarchischen Relationen von v nach u)*

4.3.6 Korrektheitskriterien für SPF-Typ-Graphen

Die bisher beschriebene formale Repräsentation von SPF-Typ-Graphen sagt aus, welche Elemente in dem Graphen vorkommen, wie die Elemente miteinander verknüpft werden können und welche Abbildungsvorschriften existieren. Auf dieser Basis ist es nun möglich, die Struktur der resultierenden Graphen durch Korrektheitskriterien weitergehend einzuschränken, so dass z. B. keine Zyklen entstehen oder Constraints zwischen Knoten gebildet werden, die über einen Pfad aus hierarchischen Relationen miteinander verbunden sind. Im Folgenden werden die Korrektheitskriterien für SPF-Typ-Graphen daher formal definiert [Reuter 2010].

Wurzelbedingungen: Jeder SPF-Typ-Graph verfügt über genau einen Wurzelknoten, der dadurch ausgezeichnet ist, dass er keinen übergeordneten Knoten besitzt. Besteht ein SPF-Typ-Graph lediglich aus einem Wurzelknoten, so handelt es sich gleichzeitig auch um einen SPF Knoten; in diesem Fall muss also die Wurzel mit einem Prozessfragment verknüpft werden. Die Kardinalität des Wurzelknotens beträgt immer *eins*.

$r \in V$ *(Existenz des Wurzelknotens)*

$\nexists v \in V: v \rightarrow r$ *(Kein übergeordneter Knoten)*

$VC(r) = 1$ *(Keine Klone des Wurzelknotens)*

$|V| = 1 \Leftrightarrow r \in V_{SPF}$ *(Wurzel als SPF-Knoten)*

SPF-Knotenbedingung: Für SPF-Knoten gilt neben der Tatsache, dass sie über keinen untergeordneten Knoten verfügen, die Bedingung, dass ihnen genau ein Prozessfragment aus der Menge der Prozessfragmente zugeordnet sein muss. Auf diese Weise wird das semantische Prozessfragment gebildet.

$\forall v \in V_{SPF} \exists! p \in P: VP(v) = p$ *(Zuordnung genau eines Fragments)*

Knotenerreichbarkeit: Jeder Knoten innerhalb des SPF-Typ-Graphen muss über einen Pfad von dem Wurzelknoten aus erreichbar sein. Dies gilt für Kontext- und SPF-Knoten gleichermaßen. Dabei ist es erlaubt, dass derselbe Knoten über hierarchische Relationen mit mehr als einem übergeordneten Knoten verbunden ist (siehe Erläuterungen zu Mehrfachreferenzen in Kapitel 4.3.3).

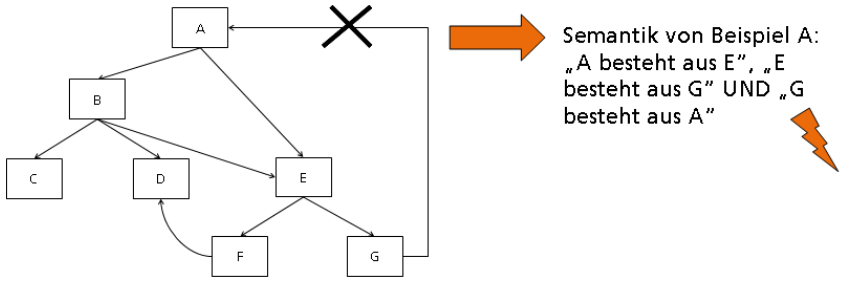
$\forall v \in V: v \neq r \Rightarrow \exists u \in V: u \rightarrow v$ *(Knotenerreichbarkeit)*

Zyklenfreiheit: SPF-Typ-Graphen sind nur dann korrekt, wenn sie mit Blick auf die hierarchischen Relationen zyklenfrei sind.

$\forall v, u \in V: v \rightarrow^* u \Rightarrow \neg(u \rightarrow^* v)$ *(Zyklenfreie hierarchische Relationen)*

Die nachfolgende Abbildung zeigt ein Beispiel für einen unerlaubten Zyklus innerhalb der hierarchischen Relationen.

Abbildung 47 Beispiel für einen unerlaubten Zyklus in SPF-Typ-Graphen



Wie leicht einzusehen ist, enthält das Beispiel einen ungültigen Zyklus zwischen den Knoten *A* und *G*. Sofern *G* Nachfolger von *A* sein soll, kann umgekehrt *A* nicht Nachfolger von *G* sein. Interessant ist die Feststellung, dass die Kante von Knoten *F* nach *D* korrekt ist, da es sich dabei nicht um einen Zyklus handelt. Nach dieser Spezifikation kann eine Konfiguration den Knoten *D* also als Nachfolger des Knotens *B* enthalten; ebenso ist es möglich, dass im SPF-Graph Knoten *D* Nachfolger von Knoten *F* ist.

Constraintbedingungen: Constraint-Relationen dürfen nur zwischen solchen Knoten definiert werden, die nicht über einen Pfad aus hierarchischen Relationen miteinander verbunden sind, da die Beziehungen zwischen über- und untergeordneten Knoten ausschließlich über die logischen Operatoren determiniert werden.

$\forall v, u \in V: (v, u, ct \in ConstraintType) \in C \Rightarrow \neg(v \rightarrow^* u \vee u \rightarrow^* v)$ *(Horizontale Constraint-Relationen)*

Bei Anforderungs- und Sequenz-Constraints muss außerdem deren Transitivität berücksichtigt werden. Erfordert ein Knoten *v* einen Knoten *u* und dieser wiederum einen Knoten *z*, dann folgt, dass es auch einen Anforderungs-Constraint von *v* nach *z* geben muss.

$\exists v, u, z \in V: (v, u, REQUIREMENT), (u, z, REQUIREMENT) \in C \Rightarrow \exists (v, z, REQUIREMENT) \in C$ *(Transitivität bei Anforderungs-Constraints)*

$\exists v, u, z \in V: (v, u, SEQUENCE), (u, z, SEQUENCE) \in C \Rightarrow \exists (v, z, SEQUENCE) \in C$ *(Transitivität bei Sequenz-Constraints*)

Darüber hinaus erstreckt sich eine Constraint-Relation immer auf den gesamten »Gültigkeitsbereich«, der aus den beiden Knoten, auf den sich der Constraint direkt bezieht, einschließlich aller ihrer untergeordneten Knoten besteht; eine Constraint-Relation der Form $(v, u, ct) \in C$ bedeutet daher, dass zwischen *v* und *u* und ihren Nachfolgern keine weiteren Constraints mehr definiert werden dürfen. Um die Constraintbedingungen formal spezifizieren zu können, wird zunächst eine Hilfsfunktion angegeben, mit der der kleinste gemeinsame Kontextknoten für eine Menge von Knoten festgestellt werden kann.

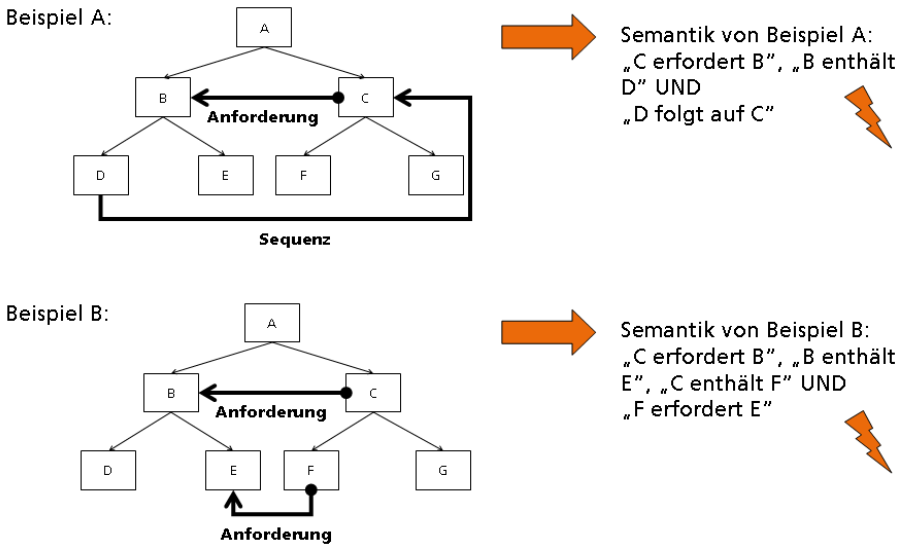
Definition (Kleinster gemeinsamer Kontextknoten). Sei *V* die Knotenmenge eines SPF-Typ-Graphen und sei $Z \subseteq V$. Ferner bezeichne 2^M die Potenzmenge einer beliebigen Menge *M*. Die Funktion *minNode* definiert für eine Menge von Knoten aus *V* den kleinsten gemeinsamen übergeordneten Knoten bzw. Kontextknoten:
$minNode: 2^V \mapsto V$
mit
$minNode(Z) = \{v \in V | v \rightarrow^* Z \setminus v \wedge \neg(\exists u \in V: u \rightarrow^* Z \wedge v \rightarrow^* u)\}$

Die folgende Constraintbedingung verhindert, dass sich unterschiedliche Constraints auf denselben Gültigkeitsbereich beziehen können:

$\forall v, u, \in V, ct_1 \in ConstraintType: (v, u, ct_1) \in C \Rightarrow \nexists x, y \in V, ct_2 \in ConstraintType: (x, y, ct_2) \in C \setminus (v, u, ct_1) \wedge ((v \rightarrow^* x \vee u \rightarrow^* x \vee x = v \vee x = u) \wedge (v \rightarrow^* y \vee u \rightarrow^* y \vee y = v \vee y = u)) \wedge (minNode(x, y) \rightarrow^* v \vee minNode(x, y) \rightarrow^* u)$ *(Einschränkung der Menge an Constraints pro Gültigkeitsbereich)*

Dieses Kriterium macht bereits die Spezifikation doppelter Constraints mit identischen Quell- und Zielknoten oder die Bildung von Schleifen unmöglich; darüber hinaus wird jedoch auch die Definition von Constraints mit demselben Gültigkeitsbereich im Hinblick auf komplette Subgraphen vermieden. Die nachfolgende Abbildung verdeutlicht an Hand von zwei Beispielen, dass die Einhaltung des Korrektheitskriteriums »ein Constraint pro Gültigkeitsbereich« für die semantische Integrität des SPF-Typ-Graphen eine wichtige Rolle spielt.

Abbildung 48 Problematik mehrerer Constraint-Relationen innerhalb desselben Gültigkeitsbereichs



In Beispiel *A* existiert zwischen den beiden Knoten *B* und *C* die Constraint-Relation »C erfordert B«; dieser Constraint bezieht sich nicht nur auf die einzelnen Knoten, sondern den gesamten Block, der durch die nachfolgenden Knoten von *B* bzw. *C* gebildet wird. D. h. auch für den Knoten *D* gilt indirekt der Constraint »C erfordert D«. Demzufolge darf kein Constraint definiert werden, der zwischen *C* und *D* eine eigene Beziehung ausdrückt. Dasselbe gilt auch für Beispiel *B*: Die Semantik des Constraints »F folgt auf E« steht zwar nicht im Widerspruch mit dem Constraint für die übergeordneten Knoten; der Constraint ist jedoch auch nicht erfor-

derlich, da »C erfordert B« bereits impliziert, dass alle Prozessfragmente, die mit *C* assoziiert sind, in der Prozessdefinition allen Fragmenten, die *B* zugeordnet sind, nachfolgen.

Schließlich gibt es auch unzulässige Kombinationen von Constraints. So darf zwischen zwei Knoten, die von demselben Knoten benötigt werden kein Ausschluss-Constraint definiert sein:

$\exists v, u, z \in V$: (v, u, REQUIREMENT) $\in C \wedge$ (v, z, REQUIREMENT) $\in C \Rightarrow \nexists$ (u, z, EXCLUSION) $\in C \wedge \nexists$ (z, u, EXCLUSION) $\in C$ (Einschränkung der Verwendung des Ausschluss-Constraints)

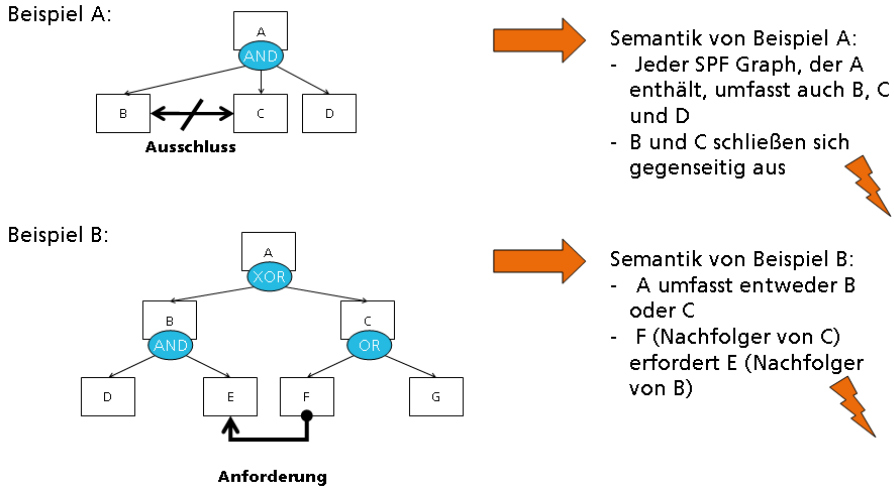
Operator-Constraint-Kombinationsbedingungen: Neben Bedingungen, die sich ausschließlich auf die Gestaltung von Constraints beziehen, müssen zusätzlich Einschränkungen im Hinblick auf mögliche Kombinationen von Operatoren und Constraint-Relationen definiert werden. So existieren Kombinationen, deren Semantik von vorneherein fehlerhaft ist und die daher in SPF-Typ-Graphen als inkorrekt erkannt werden müssen. Stehen zwei Knoten in gegenseitigem Ausschluss, dann muss der SPF-Typ-Graph solche Kombinationen von Operatoren verhindern, die dazu führen, dass in jeder Konfiguration beide Knoten enthalten sind.

$\forall v, u \in V$: (v, u, EXCLUSION) $\in C \Rightarrow \exists z \in Z$: VO(z) = OPT $\vee$ (VO(z) $\neq$ AND $\wedge$ ($\exists x \in V \setminus Z$: $z \rightarrow x$)) mit $Z = \{z \in V \mid z = minNode(v, u) \vee z = v \vee z = u \vee ((z \rightarrow^* v \vee z \rightarrow^* u) \wedge minNode(v, u) \rightarrow^* z)\}$ (Konfigurationsalternativen bei Ausschluss-Constraints)

Bei allen direkten und indirekten Nachfolgern eines Knotens mit XOR-Verknüpfung dürfen sogar überhaupt keine Constraints definiert werden (siehe Abbildung 49). Es gilt:

$\forall v, u \in V \; \exists z \in V$: $z \rightarrow^* v \wedge z \rightarrow^* u \wedge$ VO(z) = XOR $\Rightarrow \neg$ (v, u, ct $\in$ ConstraintType) $\in C$ (Keine Verwendung von Constraints in Kombination mit exklusiver Disjunktion)

Abbildung 49 Beispiele für unerlaubte Kombinationen von logischen Operatoren und Constraint-Relationen

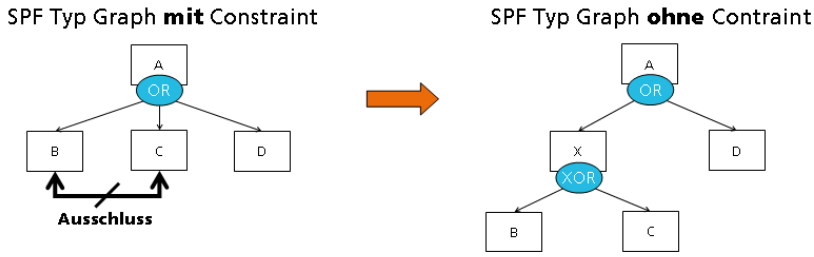


Beispiel A zeigt einen SPF-Typ-Graphen, bei dem drei Knoten *B*, *C* und *D* denselben übergeordneten Knoten *A* besitzen und für die eine AND-Verknüpfung gelten soll. Dementsprechend kann der Ausschluss-Constraint zwischen den beiden Nachfolgern *B* und *C* niemals erfüllt werden. Das zweite Beispiel illustriert zwei Teilzweige eines SPF-Typ-Graphen, zwischen denen durch den XOR-Operator eine »Entweder oder«-Logik etabliert wurde. Aufgrund dieser

Tatsache, dass jeweils nur einer der Zweige in einer Konfiguration des Graphen vorkommen kann, ist der Anforderungs-Constraint zwischen den Knoten dieser Zweige inkorrekt.

Obwohl Constraints zwischen direkten Nachfolgern eines Knoten mit Ausnahme des XOR-Operators nicht generell untersagt sind, ist es in vielen Fällen empfehlenswert, Anforderungs- und Ausschluss-Constraints zwischen Knoten mit demselben übergeordneten Kontextknoten durch geeignete Umstrukturierung des SPF-Typ-Graphen zu vermeiden. Der Grund dafür ist, dass solche Konstellationen zu Produktionsregeln in der Graphgrammatik führen können, deren Auswirkungen immer im Widerspruch zum Constraint stehen und daher durch Rekonfiguration automatisch rückgängig gemacht werden müssen (siehe Kapitel 4.4.3). Die Wahl einer adäquaten Struktur von SPF-Typ-Graphen kann solche Probleme von vorneherein verhindern. Die folgende Abbildung zeigt an einem Beispiel, wie sich ein Constraint zwischen den direkten Nachfolgern eines Knoten auf einfache Art und Weise vermeiden lässt.

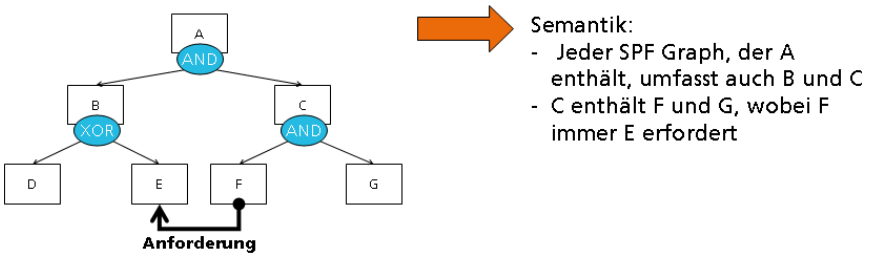
Abbildung 50 Vermeidung von Constraint-Relationen zwischen den direkten Nachfolgern desselben Kontextknotens



Wie man an dem Beispiel erkennen kann, lässt sich der Ausschluss-Constraint zwischen den benachbarten Knoten *B* und *C* durch die Einführung eines zusätzlichen Kontextknotens *X* abbilden, dem als Nachfolger *B* und *C* über einen XOR-Operator zugeordnet sind. Bei einem Anforderungs-Constraint müsste die logische XOR-Verknüpfung durch einen AND-Operator ersetzt werden und bei Bedarf zwischen *B* und *C* ein Sequenz-Constraint eingefügt werden.

Neben offensichtlich falschen Kombinationen von Operatoren und Constraints kann es mit zunehmender Komplexität des SPF-Typ-Graphen auch zu unerwünschten Konstellationen kommen, die nicht so einfach zu erkennen sind. Die nachfolgende Abbildung demonstriert an einem Beispiel, dass ein Knoten aufgrund der gewählten Operatoren und Constraints nicht ausgewählt werden kann, obwohl er im SPF-Typ-Graphen vorgesehen ist.

Abbildung 51 Problematik der Nicht-Selektierbarkeit von Knoten



Konfigurationen des SPF-Typ-Graphen aus dem Beispiel enthalten aufgrund des logischen AND-Operators des Knoten *A* immer auch die Knoten *B* und *C*; *C* umfasst zwingend die Kno-

ten *F* und *G*, wobei *F* den Einschluss des Knotens *E* als Nachfolger von *B* bedingt; theoretisch kann daher keine Konfiguration Knoten *D* enthalten. Dies ist richtig, sofern die Kardinalität von Knoten *B* und *E* jeweils eins beträgt. Da der Constraint Anforderung lediglich voraussetzt, dass mindestens ein *E* für beliebig viele Knoten *F* vorhanden ist, kann *D* gewählt werden, sobald ein *E* in der Konfiguration existiert. Sind also die Kardinalitäten richtig gesetzt, tritt dieses Problem nicht mehr auf.

Das Erkennen von solchen, gegebenenfalls problematischen Konstellationen kann durch die Spezifikation geeigneter Algorithmen automatisiert werden. Solche Algorithmen können z. B. genutzt werden, um zu evaluieren, in wie vielen Konfigurationen ein Knoten vorkommen kann. Auf diese Weise können Fehler erkannt werden, die zwar nicht zu technisch inkorrekten Prozessdefinitionen führen, aber unerwünschte Auswirkungen im Hinblick auf deren Semantik haben. Ziel dieser Arbeit ist es jedoch nicht Automatismen zu formulieren, die es ermöglichen, Knoten innerhalb von SPF-Typ-Graphen zu erkennen, die nicht selektiert werden können. Es sollen vielmehr Methoden beschrieben werden, die Fachanwender bei der Modellierung und dynamischen Adaption von Prozessen unterstützen und dabei die Entstehung inkorrekter Prozessdefinitionen verhindern.

4.3.7 Zusammenfassung der Ergebnisse

In vielen Einführungsprojekten von WfMS stellt der Umstand, dass ausführbare Geschäftsprozesse von Beginn an exakt definiert werden müssen, eine hohe Hürde bei der Entwicklung ausführbarer Prozessdefinitionen dar. Insbesondere wenn es sich um interdisziplinäre und stationsübergreifende Behandlungsabläufe handelt, ist dies keine einfache Aufgabe; einige der zu modellierenden Sachverhalte, wie z. B. die genauen Ablaufreihenfolgen und Verantwortlichkeiten, sind gar nicht explizit bekannt und müssen aufwändig erhoben werden. Mit bisherigen Ansätzen der Prozesskonfiguration wird zwar versucht, den Modellierungsaufwand generell zu senken und ein höheres Potential an Wiederverwendbarkeit von bestehenden Definitionen zu erzielen; das Problem der notwendigen Erhebung der konkreten Prozessabläufe mit all ihren Details bleibt jedoch bestehen.

Aus diesem Grund stellen SPF-Typ-Graphen ein alternatives Verfahren zur Prozessmodellierung dar. Bei SPF-Typ-Graphen handelt es sich um eine spezielle Form von Feature-Modellen, die grundsätzlich der Abbildung des Prozesswissens innerhalb einer Domäne dienen. Im Gegensatz zu anderen Lösungen im Bereich Prozesskonfiguration repräsentiert der SPF-Typ-Graph keinen speziellen prozeduralen Basisprozess, der durch Selektion von Varianten oder Einstellen einzelner Knoten so konfiguriert werden kann, dass er die Anforderungen eines ganz bestimmten Behandlungsfalls erfüllt. Die hier beschriebene Methodik macht es hingegen möglich, umfangreiches Prozesswissen in einem deklarativen Modell zu kapseln, wobei die bewährten Verfahren der Feature-Modellierung zum Einsatz kommen.

In der Definitionsphase wird der Fokus nicht auf einen einzelnen Prozess gelegt, sondern das Krankenhaus wird in operative Einheiten zerlegt, die separat voneinander betrachtet und modelliert werden können; dabei werden nur solche Aktivitäten in Form von Prozessfragmenten spezifiziert, die auch tatsächlich Relevanz bei der Umsetzung des Behandlungsprozesses als klinischer Pfad haben. Nach der getrennten Analyse der operativen Einheiten, können die entstandenen Teilbäume in einem übergeordneten SPF-Typ-Graph zusammengeführt werden. Constraint-Relationen werden nur dann definiert, wenn Beziehungen zwischen Prozessfragmenten wirklich prozessübergreifenden Allgemeinanspruch haben. Auf diese Weise wird die

Gefahr der Überspezifikation bei prozeduralen Modellierungssprachen vermieden, die zum einen einen hohen Aufwand zur Folge hat und zum anderen äußerst komplex und fehleranfällig ist. Durch die semantische Anreicherung des SPF-Typ-Graphen mit Hilfe logischer Operatoren, Constraint-Relationen und Kardinalitäten besteht jedoch die Möglichkeit das deklarative Modell schrittweise mit prozeduraler Logik zu versehen. Dieser fliesende Übergang erleichtert die Einführung von Soll-Prozessen und »Best Practices«. Anschließend bilden SPF-Typ-Graphen die Ausgangsbasis für die Ableitung konkreter Prozessdefinitionen für klinische Pfade durch Konfiguration.

Zusammenfassend konnten in diesem Kapitel die folgenden Ergebnisse erzielt werden:
- Entwicklung eines Konzepts zur Abbildung des domänenorientierten Prozesswissens durch Techniken der Feature-Modellierung
- Spezifikation von Prozessfragmenten als wieder verwendbare Basiskomponenten zur Erstellung neuer Prozessdefinitionen
- Formale, mengenbasierte Spezifikation von SPF-Typ-Graphen, deren Eigenschaften und Korrektheitskriterien

4.4 Konfigurationsphase

Die Konfigurationsphase spiegelt den Vorgang zur Erstellung einer Prozessdefinition wieder, wobei sie traditionelle Prozessmodellierungsansätze ablöst. Gemäß der konventionellen, prozeduralen Methodik werden zunächst Art und Reihenfolge der auszuführenden Aktivitäten spezifiziert sowie die Datenobjekte festgelegt, die während der Prozesslaufzeit produziert und zwischen den Aktivitäten ausgetauscht werden. Erst im Anschluss daran werden die Anwendungskomponenten identifiziert bzw. entwickelt, durch die die Aktivitäten IT-technisch realisiert werden. Dieses Verfahren bewirkt jedoch eine strikte Trennung zwischen der fachlogischen und der technischen Sicht auf Geschäftsprozesse, die letztlich in einer geteilten Verantwortung beim Entwurf und der technologischen Umsetzung von Prozessdefinitionen resultiert. Um die sich daraus ergebenden Probleme im Hinblick auf den notwendigen Abstimmungs- und Koordinationsaufwand sowie Diskrepanzen zwischen der fachlichen Semantik und der technischen Realisierung zu verhindern, wurde in dieser Arbeit ein Modell definiert, das es ermöglicht, Implementierungsbausteine zu beschreiben und im Kontext mit anderen Bausteinen anzuordnen, so dass letztlich das Abbild des Prozesswissens innerhalb einer Domäne entsteht.

SPF-Typ-Graphen entsprechen spezialisierten Feature-Modellen, mit denen die zur Verfügung stehenden technisch operationalisierten Prozessfragmente semantisch angereichert und miteinander in Beziehung gesetzt werden können. Während SPF-Typ-Graphen also das gesamte Prozesswissen repräsentieren, handelt es sich bei SPF-Graphen um Konstrukte, die konform zu den Definitionen und Festlegungen im SPF-Typ-Graphen den strukturellen Aufbau eines einzelnen Geschäftsprozesses beschreiben. SPF-Graphen entstehen während der Konfigurationsphase durch die Anwendung der Produktionsregeln einer Graphgrammatik, die aus dem SPF-Typ-Graphen automatisch abgeleitet werden kann. Die Konfigurationsphase dient damit der Entwicklung klinischer Pfade gemäß den Prinzipien der kompositionalen Prozessmodellierung. Das Vorliegen eines SPF-Graphen ist die Voraussetzung für die Generierung einer Prozessdefinition aus dem prozessorientierten Domänenwissen.

In diesem Kapitel werden SPF-Graphen als getypte Graphen formal spezifiziert; dabei wird auch gezeigt, welchen Einfluss die Deklaration logischer Verknüpfungen, Kardinalitäten, Constraints und die hierarchische Struktur der Knoten des SPF-Typ-Graphen auf den konkreten

Aufbau der SPF-Graphen hat. Ausgehend davon wird anschließend dargestellt, wie eine Graphgrammatik automatisch aus dem SPF-Typ-Graphen gewonnen werden kann und wie durch Anwendung der Graphgrammatik SPF-Graphen als Konfigurationen eines SPF-Typ-Graphen erzeugt werden können. Da Informationen zu Constraint-Relationen nicht direkt in die Graphgrammatik einfließen, wird ein Verfahren spezifiziert, nach dem Constraints während des Konfigurationsprozesses berücksichtigt werden können. Schließlich wird gezeigt, wie sich der Konfigurationsvorgang insgesamt durch den Einsatz geeigneter Automatismen beschleunigen lässt. Zuletzt werden Strategien zur manuellen Konfiguration von SPF-Typ-Graphen durch Fachanwender beschrieben.

4.4.1 SPF-Graphen als Ergebnis der Konfiguration von SPF-Typ-Graphen

SPF-Graphen repräsentieren prozessspezifische Konfigurationen von SPF-Typ-Graphen und bilden damit die Grundlage zur Ableitung einer Prozessdefinition aus dem allgemeinen Prozesswissen innerhalb einer Domäne. Während die Prozessdefinition in einer Modellierungssprache erstellt werden muss, die von dem WfMS, das die Prozessausführung übernehmen soll, interpretiert und verarbeitet werden kann, abstrahieren SPF-Graphen ähnlich wie Prozessdefinitionen in SPOT-MM von solchen systemspezifischen Details.

SPF-Graphen entstehen durch Anwendung einer Folge von Produktionsregeln, welche in einer Graphgrammatik enthalten sind, die gemäß der Spezifikation eines SPF-Typ-Graphen erzeugt worden ist. SPF-Graphen verfügen über eine baumartige Struktur, die konform zu einem SPF-Typ-Graphen gebildet wird, so dass der semantische Kontext, indem ein ausgewähltes Prozessfragment steht, zu jedem Zeitpunkt nachvollzogen werden kann. Damit unterscheiden sich SPF-Graphen von den meisten Konfigurationsansätzen im Bereich der Feature-Modellierung, bei denen die Konfiguration lediglich aus der Menge der im Feature Diagramm ausgewählten Knoten besteht. Sobald jedoch das Klonen von Knoten und der Einsatz von Mehrfachreferenzen möglich ist, ist eine graphartige Strukturierung der Konfiguration unverzichtbar [Czarnecki et al. 2005].

Definition (SPF-Graph). Bei einem SPF-Graphen handelt es sich um einen getypten Graphen, der eine prozessspezifische Konfiguration eines gegebenen SPF-Typ-Graphen repräsentiert. Ein SPF-Graph entspricht dem Tupel $SG = (V, E, V_T, V_{NT}, R, A, VA, VID)$, für das gilt:
- V (endliche Knotenmenge)
- $E \subseteq V \times V$ (endliche Menge von hierarchischen Knotenrelationen)
- $V_T \subseteq V$ (endliche Menge von terminalen Knoten)
- $V_{NT} \subseteq V$ (endliche Menge von nicht-terminalen Knoten)
- $R \subseteq V$ (endliche Menge von Wurzelknoten)
- A (endliche Menge von Knotenattributen)
- $VA: V \times A \nrightarrow \{Undefined\}$ (Attributwertfunktion)
- $VID: V \nrightarrow V^*$ (Identitätsfunktion)

Die Elemente V, E, A und VA sind analog zu SPF-Typ-Graphen definiert (siehe Kapitel 4.3.5); die Knotenmenge V darf in SPF-Graphen jedoch prinzipiell auch leer sein. Im Gegensatz zu SPF-Typ-Graphen wird bei SPF-Graphen eine Unterscheidung zwischen terminalen und nicht-terminalen Knoten benötigt. Terminale Knoten haben ein identisches Abbild im SPF-Typ-Graphen und können nicht durch die Anwendung von Produktionsregeln ersetzt werden. Nicht-terminale Knoten hingegen verfügen über keine Entsprechung im SPF-Typ-Graphen; ihre Ersetzung unter Nutzung der Produktionsregeln ist möglich. Üblicherweise besitzen SPF-

Graphen genau wie SPF-Typ-Graphen nur einen Wurzelknoten. Dennoch ist *R* als Menge definiert, da SPF-Graphen auch Bestandteile von Produktionsregeln einer Graphgrammatik sind und in dieser Funktion mehrere Wurzelknoten aufweisen können. Ist die Kantenmenge $E \neq \varnothing$, so sind die Knoten innerhalb von SPF-Graphen in einer strengen Baumstruktur angeordnet; d. h. Mehrfachreferenzen aus SPF-Typ-Graphen werden in Klone von Knoten umgewandelt, die an unterschiedlichen Stellen des Graphen vorkommen. Sei $SG = (V, V_T, V_{NT}, R, E, A, VA, VID)$ ein SPF-Graph, dann gilt:

$$\forall v, u, z \in V: v \rightarrow z \wedge u \rightarrow z \Rightarrow v = u \text{ (Einhaltung der Baumstruktur)}$$

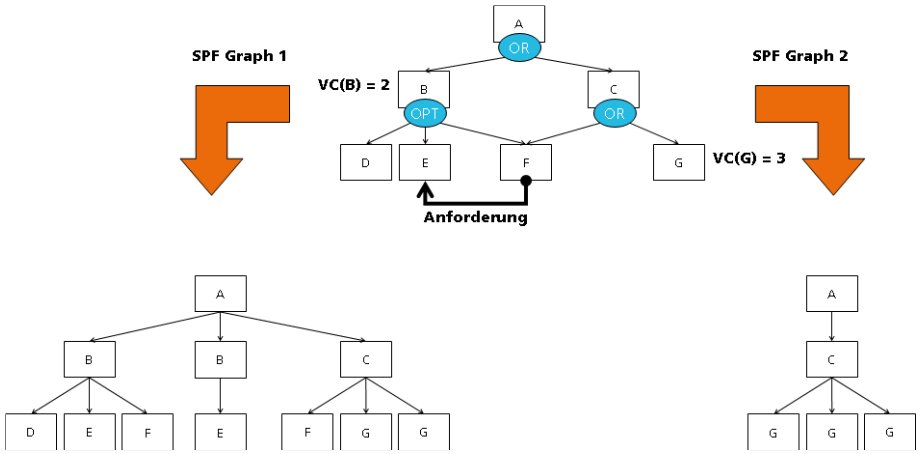
Die Identitätsfunktion *VID* ordnet jedem Knoten des SPF-Graphen eine eindeutige Knotenbezeichnung bzw. Identität aus der Menge *V** zu, wobei gilt:

$$V^* \cap V = \varnothing$$

Diese zusätzliche Identität ist notwendig, da es in SPF-Graphen anders als bei SPF-Typ-Graphen Klone desselben Knoten geben kann, wodurch es nicht mehr möglich ist, die Eindeutigkeit der aus dem SPF-Typ-Graph stammenden Knotenbezeichnungen zu gewährleisten. Durch die Identitätsfunktion können also Knoten, die gemäß ihrer Bezeichnung aus dem SPF-Typ-Graphen identisch sind, im SPF-Graphen immer noch voneinander unterschieden werden.

Die formale Spezifikation von SPF-Graphen zeigt auch, dass viele Informationen aus dem SPF-Typ-Graphen nicht übernommen werden. Während SPF-Typ-Graphen über logische Verknüpfungen, Angaben zu Kardinalitäten, Constraints und Prozessfragmentzuordnungen verfügen, enthalten SPF-Graphen diese Elemente nicht mehr. Solche Angaben beeinflussen zwar die Bildung von SPF-Graphen, sind jedoch als Information nur in SPF-Typ-Graphen enthalten. Die nachfolgende Abbildung repräsentiert zwei mögliche SPF-Graphen, die auf demselben gemeinsamen SPF-Typ-Graphen basieren. Zu beachten ist, dass in beiden SPF-Graphen alle Knoten terminalen Knoten entsprechen.

Abbildung 52 Beispiel für die Ableitung von SPF-Graphen von einem gemeinsamen SPF-Typ-Graphen



Wie bereits erwähnt, dürfen nur solche SPF-Graphen gebildet werden, die konform sind zu einem gegebenen SPF-Typ-Graphen. Um diese Konformität sicherzustellen, handelt es sich bei SPF-Graphen um »getypte Graphen«; das bedeutet, dass es zu jedem SPF-Graphen einen Graphmorphismus auf einen definierten SPF-Typ-Graphen gibt. Ein Graphmorphismus

wiederum entspricht einer Funktion, die einen Graphen als Abbild eines zweiten Graphen spezifiziert [Ehrig et al. 2006][40]. Die Typisierungsbedingung für SPF-Graphen weist jedoch eine Besonderheit auf: Sie bezieht sich lediglich auf den Teilbaum des SPF-Graphen, der durch die terminalen Knoten und ihre Kanten gebildet wird; nicht-terminale Knoten bleiben hingegen unberücksichtigt. Demzufolge existiert zwischen einem SPF-Graphen *SG* und einem SPF-Typ-Graphen *STG* ein Graphmorphismus, wenn unter Einhaltung der hierarchischen Struktur von *STG* jeder terminale Knoten von *SG*, jedes seiner Attribute und jede Kante zwischen terminalen Knoten auf einen identischen Knoten, ein identisches Attribut und eine identische Kante von *STG* abgebildet wird. Folglich darf *SG* keine terminalen Knoten, Kanten oder Attribute enthalten, die nicht zuvor in *STG* festgelegt wurden. Darüber hinaus müssen die Attribute der terminalen Knoten in *SG* dieselben Werte wie die Attribute der Knoten in *STG* annehmen, sofern dort bereits fixe Werte eingetragen worden sind.

Definition (Graphmorphismus zwischen SPF-Graphen und SPF-Typ-Graphen). Sei $SG = (V^{SG}, E^{SG}, V^{SG}_T, V^{SG}_{NT}, R^{SG}, A^{SG}, VA^{SG}, VID^{SG})$ ein SPF-Graph und $STG = (V^{STG}, E^{STG}, V^{STG}_{SPF}, r^{STG}, C^{STG}, A^{STG}, VO^{STG}, VC^{STG}, VP^{STG}, VA^{STG})$ ein SPF-Typ-Graph; die Abbildungsfunktion $f = (g_v: V^{SG}_T \mapsto V^{STG}, g_e: E^{SG} \mapsto E^{STG}, g_a: A^{SG} \mapsto A^{STG}): SG \mapsto STG$ wird Graphmorphismus genannt, wenn jeder terminale Knoten, jede Kante zwischen terminalen Knoten und jedes Attribut eines terminalen Knoten ein identisches Abbild im SPF-Typ-Graphen findet. Es muss also gelten:

- $\forall v^{SG}_T \in V^{SG}_T \Rightarrow \exists v^{STG} \in V^{STG}: g_v(v^{STG}_T) = v^{STG}$
- $\forall e^{SG} \in E^{SG}: source(e^{SG}) \in V^{SG}_T \Rightarrow g_v(source(e^{SG})) = source(g_e(e^{SG}))$
- $\forall e^{SG} \in E^{SG}: target(e^{SG}) \in V^{SG}_T \Rightarrow g_v(target(e^{SG})) = target(g_e(e^{SG}))$
- $\forall a^{SG} \in A^{SG} \exists a^{STG} \in A^{STG}: g_a(a^{SG}) = a^{STG} \wedge VA^{STG}(v^{STG}, a^{STG}) \neq Undefined \Rightarrow VA^{SG}(v^{SG}_T, a^{SG}) = VA^{STG}(g_v(v^{SG}_T), g_a(a^{SG}))$ mit $v^{STG} \in V^{STG}, v^{SG}_T \in V^{SG}_T$

Durch die Typisierung mittels Graphmorphismus lässt sich die strukturelle und attributbezogene Konformität des SPF-Graphen mit dem SPF-Typ-Graphen sicherstellen. Auf Basis der obigen Definition handelt es sich bei *SPF-Graph 1* und *SPF-Graph 2* aus Abbildung 52 um typisierte Graphen.

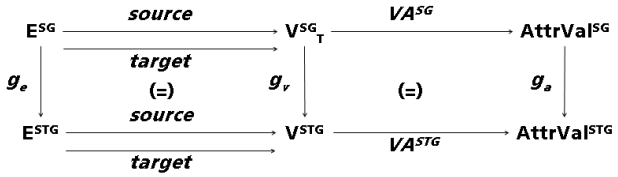
Definition (Vollständige Konfiguration eines SPF-Typ-Graphen). Ein SPF-Graph $SG = (V^{SG}, E^{SG}, V^{SG}_T, V^{SG}_{NT}, R^{SG}, A^{SG}, VA^{SG}, VID^{SG})$ ist eine »vollständige Konfiguration« eines SPF-Typ-Graphen $STG = (V^{STG}, E^{STG}, V^{STG}_{SPF}, r^{STG}, C^{STG}, A^{STG}, VO^{STG}, VC^{STG}, VP^{STG}, VA^{STG})$, wenn er keine nicht-terminalen Symbole umfasst, ein Graphmorphismus existiert, der den gesamten Graphen auf einen SPF-Typ-Graphen abbildet, und alle obligatorischen Knotenattribute eine Wertzuweisung besitzen.

- $V^{SG}_{NT} = \varnothing$
- $\exists f: SG \mapsto STG$
- $\forall v^{SG} \in V^{SG} \exists a^{SG} \in A^{SG}: AT(v^{SG}, a^{SG}) = MANDATORY \Rightarrow VA^{SG}(v^{SG}, a^{SG}) \neq Undefined$

Die nachfolgende Darstellung zeigt, welche Abbildungsvorschriften zwischen Graphelementen durch den Graphmorphismus existieren müssen. Da nicht nur die Attribute selbst, sondern auch deren Werte übereinstimmen sollen, entspricht $AttrVal^{SG}$ bzw. $AttrVal^{STG}$ der Wertemenge für Attribute von Knoten der Graphen *SG* bzw. *STG*.

[40] Siehe Ehrig et al. 2006, S. 22

Abbildung 53 Graphmorphismus als Bedingung für typisierte, vollständig konfigurierte SPF-Graphen



Neben der Typisierung als grundlegende Bedingung, die von jedem SPF-Graphen erfüllt werden muss und die die Einhaltung der hierarchischen Struktur sowie die Identität der Attribute und ihrer Attributwerte sicherstellt, wird der Aufbau der SPF-Graphen zusätzlich durch die im SPF-Typ-Graphen deklarierten Kardinalitäten, logischen Operatoren und Constraint-Relationen determiniert.

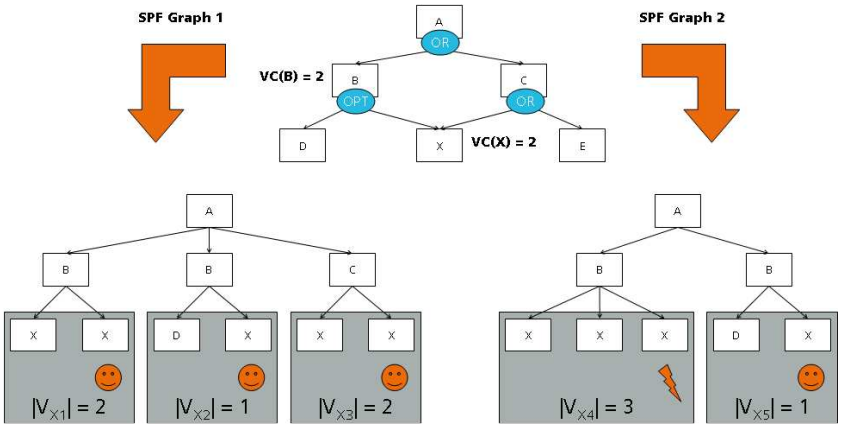
Wie in Kapitel 4.3.3 und Kapitel 4.3.5 festgelegt, verfügt jeder Knoten innerhalb eines SPF-Typ-Graphen um eine Kardinalität, die durch eine Obergrenze spezifiziert ist, die die maximale Häufigkeit repräsentiert, mit der ein Knoten im SPF-Graphen vorkommen darf. Seien $SG = (V^{SG}, E^{SG}, V^{SG}_T, V^{SG}_{NT}, R^{SG}, A^{SG}, VA^{SG}, VID^{SG})$ ein SPF-Graph und $STG = (V^{STG}, E^{STG}, V^{STG}_{SPF}, r^{STG}, C^{STG}, A^{STG}, VO^{STG}, VC^{STG}, VP^{STG}, VA^{STG})$ ein SPF-Typ-Graph; gegeben sei außerdem ein Graphmorphismus $f: SG \mapsto STG$. $V_X \subseteq V^{SG}_T$ sei eine Menge für die gilt:

$$(\exists v^{STG} \in V^{STG}: g_v(V_X) = v^{STG}) \wedge (\exists v^{SG}_T \in V^{SG}_T: v^{SG}_T \twoheadrightarrow V_X) \Rightarrow |V_X| \leq VC^{STG}(v^{STG})$$

(Kardinalitätsbedingung)

Folglich entspricht die Kardinalität eines terminalen Knotens der maximalen Anzahl für alle Klone dieses Knoten im SPF-Graphen, die über eine gerichtete Kante mit demselben übergeordneten Knoten verbunden sind und über die Funktion g_v des Graphmorphismus auf denselben Knoten im SPF-Typ-Graphen abgebildet werden. Die nachfolgende Abbildung verdeutlicht diese Bedingung an Hand von zwei Beispielen.

Abbildung 54 Auswirkung von Kardinalitäten auf die Struktur von SPF-Graphen



Die Abbildung zeigt zwei SPF-Graphen, die von demselben SPF-Typ-Graphen abgeleitet wurden. *SPF-Graph 1* hält die im SPF-Typ-Graph definierte Kardinalität für den Knoten *X* ein, wonach *X* unterhalb der übergeordneten Knoten *B* und *C* jeweils zweimal auftreten darf. *SPF-Graph 2* hingegen erfüllt die Kardinalitätsbedingung nicht, da *X* unter einem Klon von *B* insgesamt dreimal vorkommt.

Die Auswirkungen der im SPF-Typ-Graphen spezifizierten logischen Operatoren auf die Struktur der abgeleiteten SPF-Graphen hängen vom Typ der Verknüpfung ab. Während bei einer AND-Verknüpfung alle nachfolgenden Knoten im SPF-Graphen enthalten sein müssen, darf bei einer XOR-Verknüpfung nur genau ein Knoten gesetzt sein; im Falle einer OR-Verknüpfung muss mindestens ein Knoten selektiert werden und bei einer OPT-Verknüpfung können beliebig viele oder auch keiner der nachfolgenden Knoten in den Graph aufgenommen werden. Sei wiederum *SG* ein SPF-Graph, der durch einen Graphmorphismus f durch den SPF-Typ-Graphen *STG* getypt ist. Sei $v^{SG}_T \in V^{SG}_T$ ein terminaler Knoten in einem SPF-Graphen, für den mit $v^{STG} \in V^{STG}$ ein Abbild in einem SPF-Typ-Graphen existiert, so dass gilt: $g_V(v^{SG}_T) = v^{STG}$. Dann müssen in Abhängigkeit des Operators von v^{STG} die folgenden Bedingungen geprüft werden:

Wenn gilt $VO^{STG}(v^{STG}) = AND$, *dann folgt* $\forall u^{STG} \in V^{STG}: v^{STG} \rightarrow u^{STG} \Rightarrow \exists u^{SG}_T \in V^{SG}_T: g_V(u^{SG}_T) = u^{STG} \wedge g_V(source(u^{SG}_T)) = v^{STG} \wedge g_V(u^{SG}_T) \in target(v^{STG})$ *(AND-Bedingung)*

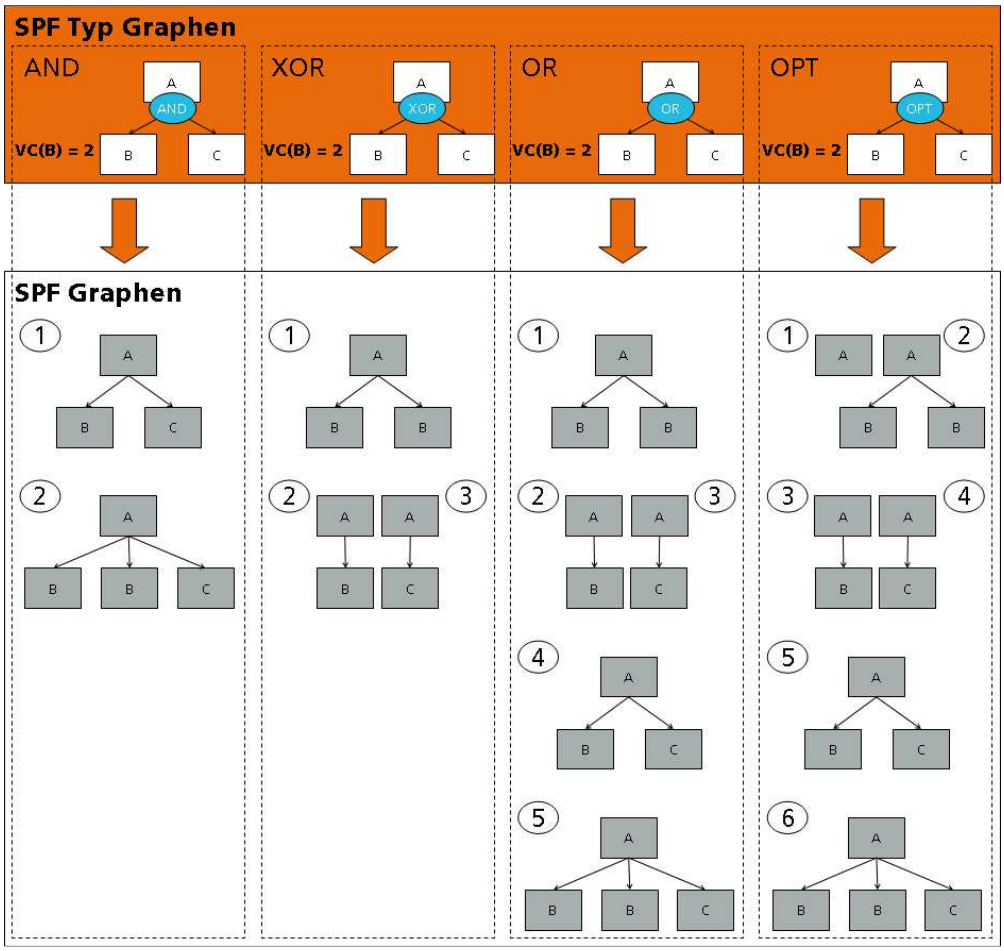
Wenn gilt $VO^{STG}(v^{STG}) = XOR$, *dann folgt* $\exists! u^{STG} \in V^{STG}: v^{STG} \rightarrow u^{STG} \Rightarrow \exists u^{SG}_T \in V^{SG}_T \wedge g_V(u^{SG}_T) = u^{STG} \wedge g_V(source(u^{SG}_T)) = v^{STG} \wedge g_V(u^{SG}_T) \in target(v^{STG})$ *(XOR-Bedingung)*

Wenn gilt $VO^{STG}(v^{STG}) = OR$, *dann folgt* $\exists u^{STG} \in V^{STG}: v^{STG} \rightarrow u^{STG} \Rightarrow \exists u^{SG}_T \in V^{SG}_T \wedge g_V(u^{SG}_T) = u^{STG} \wedge g_V(source(u^{SG}_T)) = v^{STG} \wedge g_V(u^{SG}_T) \in target(v^{STG})$ *(OR-Bedingung)*

Wenn gilt $VO^{STG}(v^{STG}) = OPT$, *dann folgt* $\forall u^{STG} \in V^{STG}: v^{STG} \rightarrow u^{STG} \Rightarrow (\exists u^{SG}_T \in V^{SG}_T: g_V(u^{SG}_T) = u^{STG} \wedge g_V(source(u^{SG}_T)) = v^{STG} \wedge g_V(u^{SG}_T) \in target(v^{STG})) \vee \neg (\exists u^{SG}_T \in V^{SG}_T: g_V(u^{SG}_T) = u^{STG} \wedge g_V(source(u^{SG}_T)) = v^{STG} \wedge g_V(u^{SG}_T) \in target(v^{STG}))$ *(OPT-Bedingung)*

Gemäß den obigen Definitionen ist es nicht ausgeschlossen, dass ein Knoten mit XOR-Operator im SPF-Graph mehrere Nachfolger hat, es handelt sich dann allerdings ausschließlich um Klone; d. h. die nachfolgenden Knoten müssen alle auf denselben Knoten im SPF-Typ-Graphen abgebildet werden. Abbildung 55 demonstriert an Hand von Beispielen, wie sich die logischen Verknüpfungen im SPF-Typ-Graphen auf den Aufbau der SPF-Graphen auswirken.

Abbildung 55 Auswirkung logischer Verknüpfungen auf die Struktur von SPF-Graphen



In der Abbildung werden alle möglichen SPF-Graphen als Ableitungen aus demselben SPF-Typ-Graphen gebildet, der lediglich in der Art der logischen Verknüpfung variiert. Wie leicht zu erkennen ist, schränken AND- und XOR-Verknüpfungen den Konfigurationsspielraum am meisten ein, während bei OR- und OPT-Verknüpfungen eine Fülle von Konfigurationsmöglichkeiten existiert.

Mit Constraint-Relationen steht das letzte Konstrukt zur Beeinflussung der Struktur von SPF-Graphen bereit. Wie bereits in Kapitel 4.3.3 erwähnt wurde, hat der Constraint »Sequenz« keine Auswirkung auf SPF-Graphen selbst, sondern ausschließlich auf die Anordnung der Prozessfragmente in Prozessdefinitionen. Im Gegensatz dazu, müssen die Constraints »Anforderung« und »Ausschluss« bereits während der Konfigurationsphase berücksichtigt werden. Um die Constraintbedingungen formal definieren zu können, wird eine Hilfsfunktion benötigt, die bereits aus Kapitel 4.3.6 bekannt ist. Die Funktion zur Feststellung des kleinsten gemeinsamen Kontextknoten wurde dort nur mit Bezug zu SPF-Typ-Graphen definiert; sie ist jedoch in analoger Weise auch auf SPF-Graphen anwendbar. $Z \subseteq V$ ist eine Teilmenge der Knotenmenge eines SPF-Typ-Graphen bzw. eines SPF-Graphen. Die Funktion *minNode:* $2^V \mapsto V$ ist folgendermaßen spezifiziert:

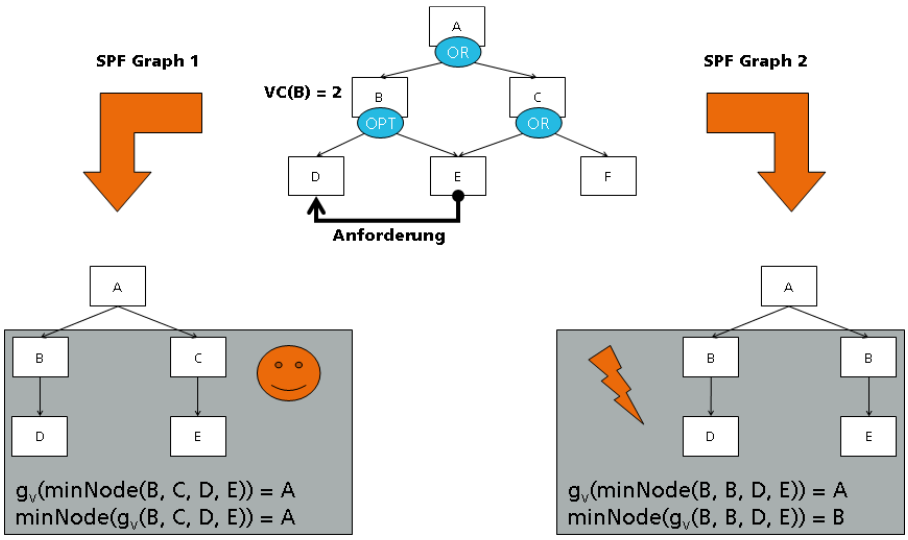
$$minNode(Z) = \{v \in V \mid v \rightarrow^* Z \setminus v \wedge \neg(\exists u \in V: u \rightarrow^* Z \wedge v \rightarrow^* u\}$$

An dieser Stelle sei darauf hingewiesen, dass der obigen Definition zufolge der kleinste gemeinsame Kontextknoten auch Bestandteil der Menge der Knoten *Z* sein kann. Sei nun *SG* ein SPF-Graph, der über einen Graphmorphismus *f* auf den SPF-Typ-Graphen *STG* abgebildet wird; seien außerdem v^{STG}, $u^{STG} \in V^{STG}$ zwei Knoten im SPF-Typ-Graphen, für die es eine Constraint-Relation $(v^{STG}, u^{STG}, REQUIREMENT) \in C^{STG}$ gibt, dann folgt:

$\forall v^{SG} \in V^{SG}: g_V(v^{SG}) = v^{STG} \Rightarrow \exists u^{SG} \in V^{SG}: g_V(u^{SG}) = u^{STG} \wedge g_V(minNode(Z^{SG})) = minNode(g_V(Z^{SG}))$ *mit* $Z^{SG} = \{z \in V^{SG} | minNode(v^{SG}, u^{SG}) \rightarrow^* z\}$ *(Anforderungsbedingung)*

Diese Definition wirkt umständlich, ist aber notwendig, wenn man das Beispiel aus Abbildung 56 betrachtet. Gegeben ist dort ein SPF-Typ-Graph, der einen Knoten *E* enthält, der sowohl von *B* als auch von *C* referenziert wird und zu dem es einen Constraint »Anforderung« auf Knoten *D* gibt. Laut Definition muss für jeden Knoten *E* innerhalb eines SPF-Graphen mindestens ein Knoten *D* unterhalb des kleinsten gemeinsamen übergeordneten Knoten existieren. In *SPF-Graph 1* entspricht die Menge Z^{SG} den Elementen *B*, *C*, *D* und *E*, deren kleinster gemeinsamer Kontextknoten *A* ist. Auch wenn man dieselbe Knotenmenge zuerst auf ihre Pendants im SPF-Typ-Graphen abbildet und anschließend den kleinsten gemeinsamen Kontextknoten bestimmt, ist *A* das Ergebnis. Demnach handelt es sich bei *SPF-Graph 1* um einen korrekten Graphen mit Hinblick auf die Constraint-Relation. Anders hingegen verhält es sich mit *SPF-Graph 2*. Der kleinste gemeinsame Kontextknoten für die Elemente *B*, *B*, *D*, *E* der Menge Z^{SG} ist *A*. Die Suche nach dem kleinsten gemeinsamen Knoten im SPF-Typ-Graphen ergibt für die Knoten *B*, *D* und *E* stattdessen *B*. Aus diesem Grund kann es sich bei *SPF-Graph 2* nicht um einen gültigen Graphen handeln; der Graph wäre jedoch korrekt, wenn die Knoten *D* und *E* Nachfolger desselben Kontextknotens *B* wären.

Abbildung 56 Auswirkung des Constraints »Anforderung« auf SPF-Graphen

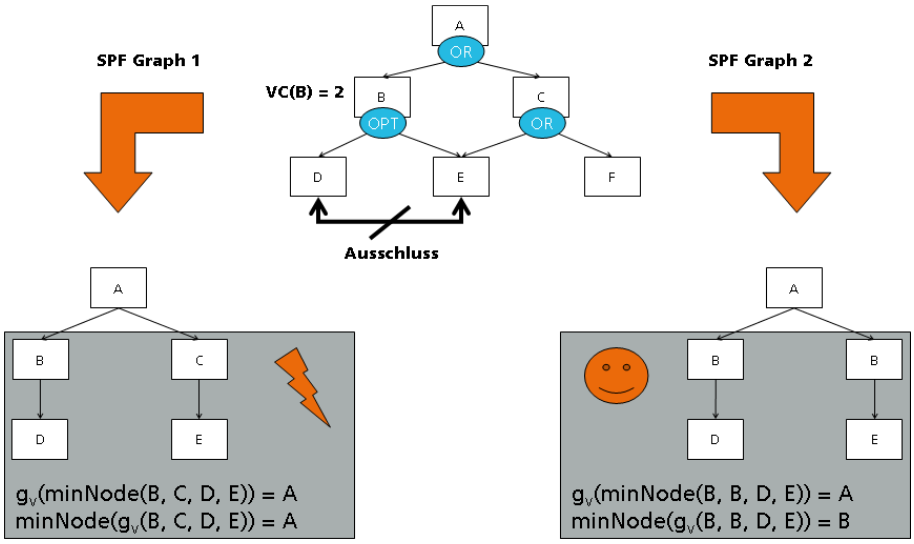


Bei dem zweiten Constraint, dessen Einfluss auf SPF-Graphen berücksichtigt werden muss, handelt es sich um die »Ausschluss«-Bedingung. Es gelten dieselben Grundannahmen wie bei dem Constraint »Anforderung«. Wenn also gilt $(v^{STG}, u^{STG}, EXCLUSION) \in C^{STG}$, dann folgt:

$\forall v^{SG} \in V^{SG}: g_v(v^{SG}) = v^{STG} \Rightarrow \neg \exists u^{SG} \in V^{SG}: g_v(u^{SG}) = u^{STG} \wedge g_v(minNode(Z^{SG})) = minNode(g_v(Z^{SG}))$ *mit* $Z^{SG} = \{z \in V^{SG} | minNode(v^{SG}, u^{SG}) \rightarrow^* z\}$[41] *(Ausschlussbedingung)*

Entsprechend dieser Definition darf von zwei Knoten, zwischen denen die Ausschlussbedingung gilt, unterhalb des kleinsten gemeinsamen Kontextknoten stets nur einer der Knoten auftreten. Damit entspricht die Ausschlussbedingung also der Negation der Anforderungsbedingung (siehe Abbildung 57).

Abbildung 57 Auswirkung des Constraints »Ausschluss« auf SPF-Graphen



In *SPF-Graph 1* aus dem obigen Beispiel kommen die sich gegenseitig ausschließenden Knoten *D* und *E* unterhalb desselben kleinsten gemeinsamen Kontextknoten vor; der Graph ist daher ungültig. *SPF-Graph 2* ist nun korrekt, da im Kontext von *B* immer wieder neu entschieden werden kann, ob *D* oder *E* gewählt wird.

4.4.2 Graphgrammatiken auf Basis von SPF-Typ-Graphen

Wie bereits an mehreren Stellen dieser Arbeit erwähnt wurde, soll die Konfiguration von SPF-Typ-Graphen und damit die Entwicklung von SPF-Graphen durch den Einsatz von Graphgrammatiken realisiert werden. Der größte Vorteil bei der Generierung und Nutzung von Graphgrammatiken liegt darin, dass das schon in ADEPT2 verfolgte Prinzip des »Correctness by Construction« auf die Bildung von SPF-Graphen übertragen werden kann. »Correctness by Construction« bedeutet hier, dass in Abhängigkeit der aktuellen Konfiguration nur eine begrenzte Anzahl von Produktionsregeln bereitgestellt wird, die zum Aufbau von SPF-Graphen eingesetzt werden dürfen, so dass die Entstehung inkorrekter Graphen automatisch verhindert

[41] Da es sich um einen Constraint mit Gegenseitigkeit handelt, gilt dieselbe Bedingung auch für alle $u^{SG} \in V^{SG}$, die auf $u^{STG} \in V^{STG}$ abgebildet werden.

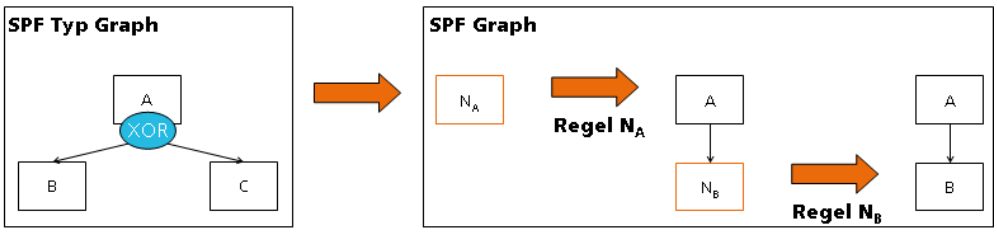
werden kann. Wie in Kapitel 4.4.3 gezeigt wird, ist durch die optionale Spezifikation von Constraints im SPF-Typ-Graphen die Anwendbarkeit des »Correctness by Construction« Prinzips jedoch nicht mehr uneingeschränkt möglich, so dass zusätzlich Algorithmen benötigt werden, um die Einhaltung der Constraints zu gewährleisten.

Bei Graphgrammatiken handelt es sich um die Übertragung des Konzepts kontextfreier Grammatiken von Worten auf Graphen. Kontextfreie Grammatiken nach Chomsky repräsentieren ein System $G = (N, T, P, S)$ mit den folgenden Komponenten:
- N ist eine nicht-leere, endliche Menge von Nichtterminalsymbolen
- T ist eine nicht-leere, endliche Menge von Terminalsymbolen mit $N \cap T = \varnothing$
- PR ist eine nicht-leere, endliche Menge von Produktionsregeln der Form $X \rightarrow Y$, wobei X ein nicht-terminales Symbol ist und Y eine geordnete Folge von Symbolen aus der Menge von Symbolen $(T \cup N)^*$
- S entspricht dem Startsymbol mit $S \in N$

Ausgehend von dem Startsymbol werden solange Produktionsregeln angewendet, bis ein Wort nur noch terminale Symbole umfasst. Dies setzt jedoch voraus, dass für die linke Seite der Produktionsregel ein entsprechendes Abbild in dem gegebenen Wort gefunden wird; anschließend wird das identifizierte Muster durch die Symbole auf der rechten Seite ersetzt. Als Beispiel sei an dieser Stelle eine Grammatik mit den folgenden vier Produktionsregeln gegeben:
- $S \rightarrow AB$
- $A \rightarrow aA \mid a$
- $B \rightarrow Bb \mid b$

Großbuchstaben repräsentieren in diesem Beispiel die Nichtterminalsymbole, Kleinbuchstaben entsprechen terminalen Symbolen. Durch die Regelanwendung entstehen Ausdrücke, die minimal aus »ab« bestehen. Die Grammatik entspricht also der Formel $a^n b^m$ (mit $n, m \in \mathbb{N} \setminus 0$). Dasselbe Prinzip lässt sich auch zur Bildung von SPF-Graphen nutzen. Da SPF-Graphen durch Anwendung von Produktionsregeln entstehen, entspricht ihre Knotenmenge einem Satz von terminalen und nicht-terminalen Symbolen, wobei es sich bei nicht-terminalen Symbolen ausschließlich um Blattknoten handeln kann. Während terminale Symbole mittels Graphmorphismus auf die Knoten desselben SPF-Typ-Graphen abgebildet werden können, lassen sich Knoten, die Nichtterminalsymbole widerspiegeln, schrittweise durch Regelanwendung ersetzen. Die folgende Abbildung verdeutlicht diesen Zusammenhang.

Abbildung 58 Prinzip der Entwicklung von SPF-Graphen durch Anwendung von Produktionsregeln



Bei den Knoten N_A und N_B handelt es sich um nicht-terminale Knoten. Alle übrigen Knoten des SPF-Graphen entsprechen terminalen Knoten. N_A ist zugleich das Startsymbol der Graphgrammatik, das durch die Anwendung der entsprechenden Regel in eine Struktur überführt wird, die aus dem terminalen Knoten A und dem untergeordneten nicht-terminalen Knoten N_B

besteht. Durch eine entsprechende Regel kann N_B schließlich in den terminalen Knoten *B* umgewandelt werden. Nach dieser Regelanwendung ist der SPF-Graph vollständig konfiguriert, weil er keine nicht-terminalen Knoten mehr enthält und ein Graphmorphismus auf den SPF-Typ-Graphen existiert.

Da die Konfiguration des SPF-Typ-Graphen durch die Auswahl und Anwendung von Produktionsregeln in mehreren Schritten erfolgt, entstehen auch mehrere Versionen von SPF-Graphen, die im Hinblick auf die Einhaltung von logischen Operatoren und Constraints noch unvollständig bzw. inkorrekt sein können. Zwischen jeder Version eines SPF-Graphen muss jedoch ein getypter Graphmorphismus existieren. Dies ist die Voraussetzung für eine korrekte Ableitung von SPF-Graphen aus einem SPF-Typ-Graphen in einem iterativen Prozess während der Konfiguration.

Definition (Getypter Graphmorphismus zwischen SPF-Graphen). Jeder SPF-Graph $SG = (V^{SG}, E^{SG}, V^{SG}_T, V^{SG}_{NT}, R^{SG}, A^{SG}, VA^{SG}, VID^{SG})$, der durch einen Graphmorphismus $f: SG \mapsto STG$ getypt ist, wird durch das Tupel (SG, f) definiert. Ein getypter Graphmorphismus bildet zwei SPF-Graphen *SG* und *SG** so aufeinander ab, dass die Abbildung f der beiden Graphen auf den SPF-Typ-Graphen *STG* erhalten bleibt. Ein getypter Graphmorphismus ist definiert als $\varphi: (SG, f) \mapsto (SG^*, f^*)$, mit:
- $\varphi = (g_v: V^{SG} \mapsto V^{SG^*}, g_e: E^{SG} \mapsto E^{SG^*}, g_a: A^{SG} \mapsto A^{SG^*}): SG \mapsto SG^*$
- $f^* \circ \varphi = f$

An die Produktionsregeln einer Graphgrammatik für den SPF-Typ-Graphen stellt sich die Anforderung, dass sie getypten Graphmorphismen entsprechen. Demnach darf keine Regel definiert werden, die dazu führt, dass der resultierende SPF-Graph nicht konform zu dem gegebenen SPF-Typ-Graphen ist. Auf diese Weise dient der SPF-Typ-Graph auch der Sicherstellung der Kompatibilität der SPF-Graphen vor und nach der Anwendung einer Produktionsregel. Die Graphgrammatik besteht also nicht nur aus der Menge der Produktionsregeln, sondern auch aus dem ursprünglichen SPF-Typ-Graphen.

Definition (Graphgrammatik auf Basis von SPF-Typ-Graphen). Eine Graphgrammatik ist ein Tupel *GG = (STG, PR, START)* mit folgenden Elementen:
- *STG (SPF-Typ-Graph, STG = (V^{STG}, V^{STG}_{SPF}, ...))*
- *PR ⊆ V x SG x SG (endliche Menge von Produktionsregeln*)
- *START (initialer SPF-Graph)*

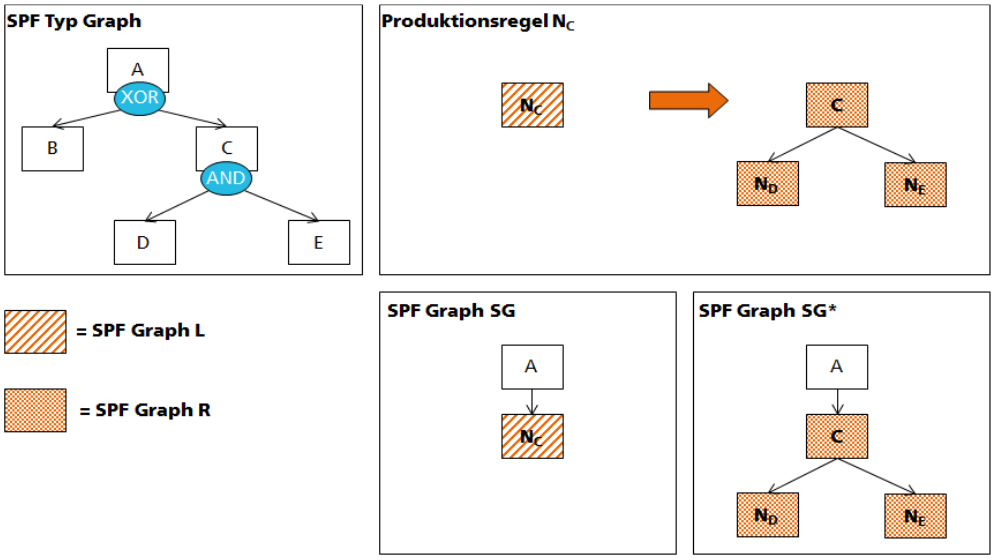
Bei *START* handelt es sich um einen SPF-Graphen, der aus genau einem nicht-terminalen Knoten besteht; der initiale SPF-Graph entspricht damit dem Startsymbol einer kontextfreien Grammatik. Eine Produktionsregel $pr = (v, L, R) \in PR$ besteht aus einer Knotenbezeichnung *v* und einem Paar von SPF-Graphen *L* und *R*. Die Transformation eines SPF-Graphen *SG* in einen SPF-Graphen *SG** an Hand der Regel *pr* ist definiert durch einen Graphmorphismus $o: L \cup R \rightarrow SG \cup SG^*$, wobei gilt:
- $o(L) \subseteq SG$ und $o(R) \subseteq SG^*$:⇔ die linke Seite der Produktionsregel ist Bestandteil des ursprünglichen SPF-Graphen *SG* und die rechte Seite der Regel ist enthalten in dem resultierenden SPF-Graphen *SG**.
- $o(L \setminus R) = SG \setminus SG^*$ und $o(R \setminus L) = SG^* \setminus SG$:⇔ aus *SG* werden exakt die Elemente aus L entfernt, die nicht in R existieren, während *SG** über die Elemente aus *R* verfügt, die nicht in *L* vorhanden sind.

Generische Ansätze im Hinblick auf Graphtransformationssysteme, sehen für jede Produktionsregel zusätzlich eine Einbettungsregel vor [Andries et al. 1999, Blockeel & Nijssen 2008]. Diese Einbettungsregel besagt, in welcher Weise mit den Kanten umgegangen werden soll, die nach dem Entfernen von $L \setminus R$ aus SG über keinen Quell- bzw. Zielknoten mehr verfügen. Eine solche Einbettungsregel als variabler Parameter jeder Produktionsregel wird in dieser Arbeit nicht benötigt, da es keine Variationen bei der Verknüpfung von Kanten aus SG^* mit den Elementen aus R gibt. Stattdessen erhält die Kante, die zuvor den Wurzelknoten von L mit seinem übergeordneten Kontextknoten verband, alle Wurzelknoten von R als Zielknoten. Sei $\varphi = (g_v: V^{SG} \nrightarrow V^{SG'}, g_e: E^{SG} \nrightarrow E^{SG'}, g_a: A^{SG} \nrightarrow A^{SG'}): SG \nrightarrow SG^*$ ein Graphmorphismus zwischen SG und SG^*. Dann gelten grundsätzlich für alle Produktionsregeln die folgenden Einbettungsregeln:

- $E^{SG^*} := E^{SG^*} \setminus \{ (x, y) \mid y = \varnothing \} :\Leftrightarrow$ alle Kanten ohne Zielknoten werden aus SG* entfernt.
- $target(\varphi(source(o(R^L)))) := o(R^R) :\Leftrightarrow$ Ziel des Vorgängers des Abbilds der Wurzel von L (nämlich R^L) in SG ist nun das Abbilds der Wurzel von R (nämlich R^R) in SG^*.

Die Zusammenhänge zwischen L, R, SG und SG^* werden an Hand eines Beispiels in der nachfolgenden Abbildung anschaulich dargestellt.

Abbildung 59 Struktur und Funktionsweise von Produktionsregeln zur Manipulation von SPF-Graphen



Wie aus diesem Beispiel hervorgeht, besteht der SPF-Graph der linken Seite einer Produktionsregel stets aus einem einzigen nicht-terminalen Knoten; der SPF-Graph der rechten Seite hingegen kann sich aus mehreren Knoten zusammensetzen, wobei gilt, dass die Wurzel immer ein terminaler Knoten ist, dem beliebig viele nicht-terminale Knoten als Nachfolger zugeordnet sind. Die Anwendung der Regel bedeutet, dass derjenige Knoten im SPF-Graphen, der den nicht-terminalen Knoten auf der linken Seite der Regel repräsentiert, durch den Graphen auf der rechten Seite der Regel ersetzt wird.

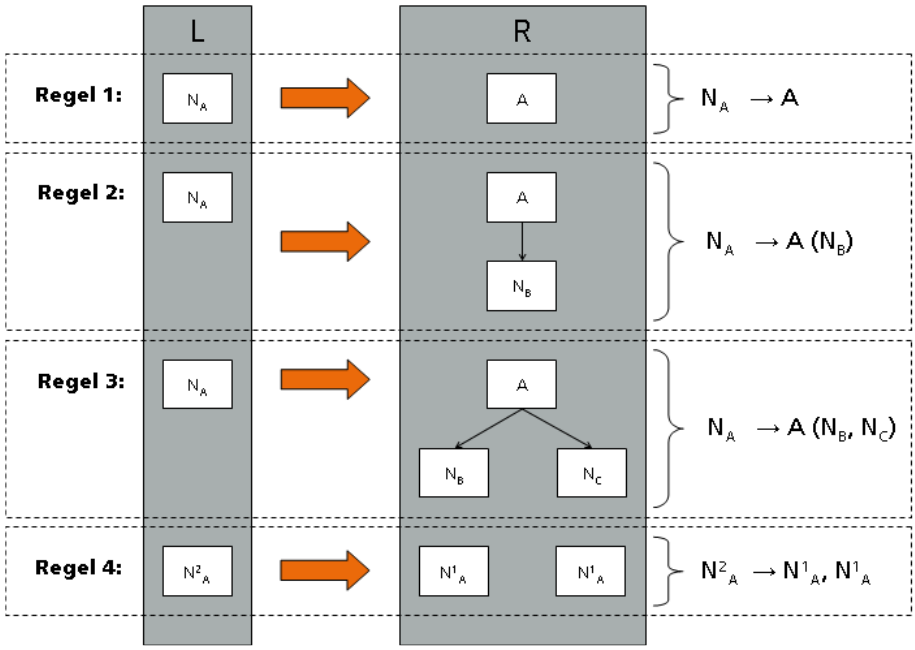
In den folgenden Kapiteln wird spezifiziert, wie aus einem SPF-Typ-Graphen automatisch die Regelmenge der Graphgrammatik generiert werden kann. Außerdem wird ein Algorithmus zur Anwendung einer Regel auf einen SPF-Graphen in Pseudocode angegeben; darüber hinaus

werden Vorgehensweisen beschrieben, die erheblich zur Reduktion des Aufwands bei der Konfiguration von SPF-Typ-Graphen beitragen.

4.4.2.1 Algorithmus zur Ableitung von Produktionsregeln aus SPF-Typ-Graphen

In diesem Kapitel wird nun ein Algorithmus zur automatischen Erzeugung von Produktionsregeln definiert. Jede Produktionsregel verfügt über eine Bezeichnung *v*, die einer Knotenbezeichnung im SPF-Typ-Graphen entspricht, sowie einen SPF-Graphen auf der linken und auf der rechten Seite. Während der Graph auf der linken Seite immer nur einen einzigen nicht terminalen Knoten umfasst, entspricht der Graph auf der rechten Seite einem Baum, der mindestens einen terminalen Wurzelknoten hat. Um die Regelmenge zu einem Graphen übersichtlich darstellen zu können, wird in den folgenden Beispielen eine vereinfachende Syntaxrepräsentation genutzt. Abbildung 60 zeigt, wie Regeln auf eine entsprechende Syntaxrepräsentation abgebildet werden können.

Abbildung 60 Syntaktische Repräsentation zur kompakten Darstellung von Produktionsregeln



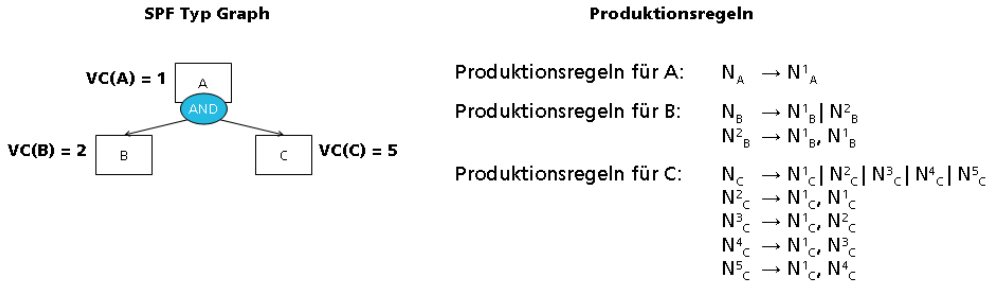
Der Ausdruck »N_X« wurde bereits in allen Beispielen genutzt, um nicht-terminale Knoten zu symbolisieren, wobei Index »X« der eindeutigen Knotenbezeichnung aus dem SPF-Typ-Graphen entspricht. Terminale Knoten hingegen werden durch die Knotenbezeichnung repräsentiert. Zu beachten ist, dass das terminale Symbol in sämtlichen Beispielen lediglich die Bezeichnung des Knotens widerspiegelt; tatsächlich setzt sich das Symbol jedoch aus dem Knoten, seinen Attributen und Attributwerten zusammen. Die Beziehung zwischen über- und untergeordneten Knoten wird durch einen Klammerausdruck deutlich gemacht; die Knotenbezeichnung des übergeordneten Knoten steht vor der Klammer, alle seine Nachfolger werden durch Komma voneinander getrennt, in der Klammer aufgelistet. Wie man am Beispiel von Regel 4 erkennen kann, könnten SPF-Graphen als rechte Seite einer Produktionsregel auch aus mehreren nicht-terminalen Wurzelknoten bestehen. Um Knotenkardinalitäten adäquat zu

berücksichtigen ist daher eine Indexierung der Nichtterminalsymbole erforderlich. Im Folgenden wird zunächst informal beschrieben, nach welchen Regeln die SPF-Graphen auf der rechten Seite gebildet werden, bevor anschließend der eigentliche Generierungsalgorithmus in Pseudocode spezifiziert wird.

Produktionsregeln für Knotenkardinalitäten

Die Umwandlung des Nichtterminalsymbols erfolgt in Abhängigkeit der Knotenkardinalität. So wird automatisch eine Menge von Nichtterminalsymbolen erzeugt, deren Anzahl exakt die Kardinalität widerspiegelt und die sich durch eine erweiterte Indexierung voneinander unterscheiden. Bei der Konfiguration des Nichtterminalsymbols des Knotens kann daher direkt entschieden werden, wie viele Klone gebildet werden sollen. Abbildung 61 zeigt ein Beispiel, wie die Knotenkardinalität bei der Erzeugung der Produktionsregeln berücksichtigt wird.

Abbildung 61 Umsetzung von Knotenkardinalitäten durch Produktionsregeln

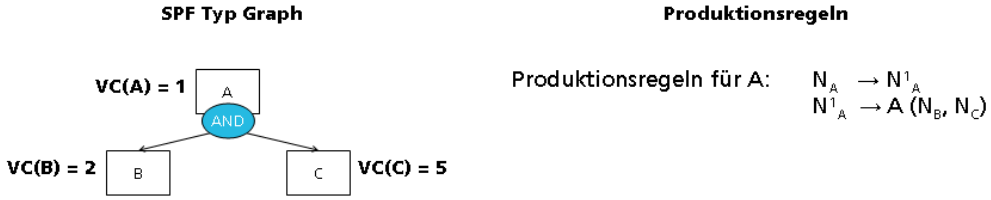


Da im Gegensatz zu dem Vorschlag von [Czarnecki et al. 2005] keine unendlichen oder komplexen Intervalle bei der Angabe einer Knotenkardinalität betrachtet werden müssen, gestaltet sich die Umsetzung der Kardinalität durch Regeln der Graphgrammatik einfacher. In dem obigen Beispiel hat Knoten *A* die Kardinalität *eins*; aus diesem Grund kann das Nichtterminalsymbol ausschließlich in exakt ein anderes Nichtterminalsymbol umgewandelt werden. Bei Knoten *B* hingegen stehen zwei mögliche Nichtterminalsymbole und bei Knoten *C* sogar fünf Symbole zur Auswahl bereit. Symbole, die Alternativen zueinander bilden, können in der syntaktischen Schreibweise durch das Zeichen »|« voneinander getrennt werden; der Ausdruck $N_C \rightarrow N^1_C \mid N^2_C$ entspricht der verkürzten Schreibweise für die zwei Regeln $N_C \rightarrow N^1_C$ und $N_C \rightarrow N^2_C$. Jedes nicht-terminale Symbol, das eine Knotenkardinalität repräsentiert, wird in das Nichtterminalsymbol für die Kardinalität eins und in das Nichtterminalsymbol mit der nächst kleineren Kardinalität umgewandelt. Auf diese Weise kann durch Anwendung der entsprechenden Produktionsregeln der Ausdruck $N^5_C \rightarrow N^1_C, N^4_C$ schließlich nach $N^5_C \rightarrow N^1_C, N^1_C, N^1_C, N^1_C, N^1_C$ transformiert werden. Bei den Produktionsregeln, die lediglich der Auflösung von Kardinalitäten dienen, tritt die Besonderheit auf, dass der Graph auf der rechten Seite der Regel keinen Baum repräsentiert, sondern stattdessen mehrere Wurzelknoten umfasst (siehe Regel 4 in Abbildung 60). Nach der Erzeugung der Regeln für die Knotenkardinalität, muss abschließend noch eine Regel generiert werden, die das Nichtterminalsymbol mit dem Kardinalitätsindex *eins* in einen Ausdruck umwandelt, dessen Struktur sich nach dem logischen Operator richtet, der dem jeweiligen Knoten zugeordnet ist.

Produktionsregeln für den AND-Operator

Neben der Produktionsregel für die Kardinalität, muss für einen Knoten, der mit einem AND-Operator verknüpft ist, lediglich eine weitere Produktionsregel angelegt werden. Die rechte Seite der Produktionsregel besteht aus dem terminalen Knoten sowie je einem nicht-terminalen Knoten für jeden Nachfolger des terminalen Knotens im SPF-Typ-Graphen; das bedeutet, dass nach der Anwendung dieser Produktionsregel, anschließend die Regeln für alle nachfolgenden Knoten ausgeführt werden können.

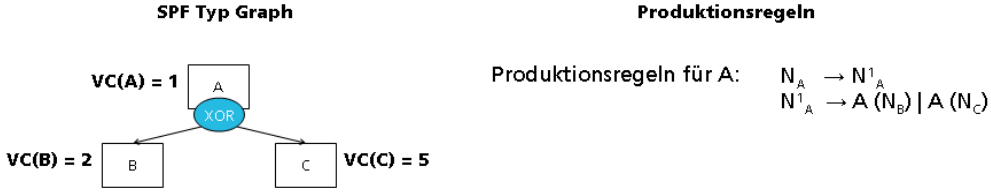
Abbildung 62 Umsetzung des AND-Operators durch Produktionsregeln



Produktionsregeln für den XOR-Operator

Im Gegensatz zu Knoten mit AND-Operator, muss für Knoten mit XOR-Operator für jeden nachfolgenden Knoten eine Produktionsregel erzeugt werden. Auf der linken Seite der Regel befindet sich jedoch stets derselbe nicht-terminale Knoten mit Knotenkardinalität eins.

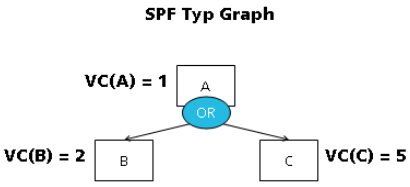
Abbildung 63 Umsetzung des XOR-Operators durch Produktionsregeln



Produktionsregeln für den OR-Operator

Für die Umsetzung des OR-Operators müssen Produktionsregeln erzeugt werden, die alle möglichen Kombinationen aus terminalem und nicht-terminalen Knoten als Alternativen zueinander bereitstellen; die grundlegende Anforderung ist, dass mindestens ein Nachfolger im SPF-Typ-Graphen durch die Selektion des entsprechenden Nichtterminalsymbols im SPF-Graphen gesetzt wird.

Abbildung 64 Umsetzung des OR-Operators durch Produktionsregeln



Produktionsregeln

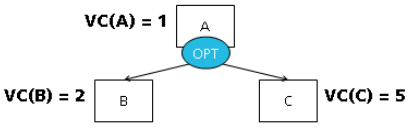
Produktionsregeln für A:

$N_A \rightarrow N^1_A$
$N^1_A \rightarrow A(N_B, N_C) \mid A(N_B) \mid A(N_C)$

Produktionsregeln für den OPT-Operator

Die Produktionsregeln für den OPT-Operator entsprechen den Regeln für den OR-Operator mit der Ausnahme, dass die rechte Seite auch ausschließlich aus dem terminalen Knoten bestehen darf, bei dem es sich im SPF-Typ-Graphen um einen Kontextknoten handelt. SPF-Knoten, die über keine Nachfolger verfügen, werden in derselben Art und Weise auf einen terminalen Knoten im SPF-Graph abgebildet.

Abbildung 65 Umsetzung des OPT-Operators durch Produktionsregeln



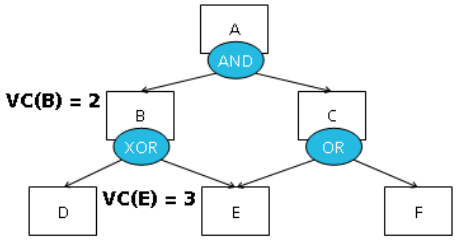
Produktionsregeln

Produktionsregeln für A:

$N_A \rightarrow N^1_A$
$N^1_A \rightarrow A(N_B, N_C) \mid A(N_B) \mid A(N_C) \mid A$

Die nachfolgende Abbildung zeigt beispielhaft die Syntaxrepräsentation für die komplette Regelmenge zu einem gegebenen Graphen.

Abbildung 66 Beispiel für eine Graphgrammatik



Produktionsregeln

$N_A \rightarrow N^1_A$
$N^1_A \rightarrow A(N_B, N_C)$

$N_B \rightarrow N^1_B \mid N^2_B$
$N^1_B \rightarrow B(N_D) \mid B(N_E)$
$N^2_B \rightarrow N^1_B, N^1_B$

$N_C \rightarrow N^1_C$
$N^1_C \rightarrow C(N_E, N_F) \mid C(N_E) \mid C(N_F)$

$N_D \rightarrow N^1_D$
$N^1_D \rightarrow D$

$N_E \rightarrow N^1_E \mid N^2_E \mid N^3_E$
$N^1_E \rightarrow E$
$N^2_B \rightarrow N^1_E, N^1_E$
$N^3_B \rightarrow N^1_E, N^2_E$

$N_F \rightarrow N^1_F$
$N^1_F \rightarrow F$

Gemäß dieser Menge von Produktionsregeln handelt es sich bei dem nachfolgenden Ausdruck um eine korrekte syntaktische Repräsentation eines SPF-Graphen:

A (B (E, E, E), C (E, E, F))

Der Algorithmus zur automatischen Generierung von Produktionsregeln für SPF-Typ-Graphen besteht aus einer großen rekursiven Funktion *generateRules*, die als Parameter eine leere Graphgrammatik, einen SPF-Typ-Graphen und einen Bool'schen Wert entgegennimmt, um daraus die Regelmenge zu erzeugen. Der Bool'sche Wert gibt an, ob die Funktion das erste Mal aufgerufen wird oder ob es sich um eine Rekursion handelt; d. h. beim Start der Funktion muss der Parameter den Wert »true« annehmen.

Formel 1 Hauptalgorithmus zur rekursiven Erzeugung von Produktionsregeln

```
generateRules(GG, STG, firstCall) {
//GG = (STG, PR, S)
//STG = (V^STG, E^STG, V^STG_SPF, r^STG, C^STG, A^STG, VO^STG, VC^STG, VP^STG, VA^STG)
      v ≔ r^STG;
      succNodes ≔ {u ∈ V^STG | v → u}; // Direkte Nachfolger
      IF (v.visited = false)
         v.visited ≔ true; // Knoten als besucht markieren
         L ≔ (N_v, ∅ , ∅ , N_v, N_v, ∅ , VA, VID); // Linker SPF-Graph
         R ≔ (N¹_v, ∅ , ∅ , N¹_v, N¹_v, ∅ , VA, VID); // Rechter SPF-Graph
         rule ≔ (v, L, R); // Regel: N_v → N¹_v
         IF (firstCall = true)
            GG ≔ (STG, rule, L); // Graphgrammatik initialisieren
         END-IF
         ELSE
            PR^GG ≔ PR^GG ∪ rule;
         END ELSE
         generateRules_CAR(GG, v, VC^STG(v)); // Regeln für Kardinalitäten
         IF (|succNodes| = 0) // Regel bildet auf terminalen Knoten ab
            L ≔ R;
            R ≔ (v, ∅ , v, ∅ , v, A^STG, VA^STG, VID); // Übernahme der Attribute
aus dem SPF-Typ-Graphen
            rule ≔ (v, L, R); // Regel: N¹_v → [v, attr]
            PR^GG ≔ PR^GG ∪ rule;
         END-IF
         ELSE
            IF (VO^STG(v) = AND)
               generateRules_AND(GG, v, succNodes, A^STG, VA^STG);
            END-IF
            ELSE IF (VO^STG(v) = XOR)
               generateRules_XOR(GG, v, succNodes, A^STG, VA^STG);
            END-ELSE-IF
            ELSE IF (VO^STG(v) = OR)
               generateRules_OR(GG, v, succNodes, A^STG, VA^STG);
            END-ELSE-IF
            ELSE
               generateRules_OPT(GG, v, succNodes, A^STG, VA^STG);
            END-ELSE
            FOR (i ≔ 0; i < |succNodes|; i + 1) // Rekursiver Aufruf
                v_i ≔ succNodes[i];
                succNodes_i ≔ {u ∈ V^STG|v_i →* u};
                V_SPFi ≔ {u ∈ V^STG_SPF|u = v_i ∨ v_i →* u};
                E_i ≔ {(u, z) ∈ E^STG| u = v_i ∨ v_i →* u};
                STG_i ≔ (v_i∪succNodes_i, E_i, V_SPFi, v_i, C^STG, A^STG, VO^STG, VC^STG, VP^STG,
VA^STG);
                generateRules(GG, STG_i, false);
            END-FOR
```

```
        END-ELSE
    END-IF
}
```

Bei dem Knoten, der aktuell behandelt werden soll, handelt es sich immer um den Wurzelknoten des übergebenen SPF-Typ-Graphen. Zunächst ermittelt der Algorithmus die Nachfolger dieses Knotens. Anschließend überprüft er, ob der Knoten im Rahmen eines rekursiven Methodenaufrufs besucht worden ist und daher bereits Produktionsregeln zu diesem Knoten existieren; dieser Fall tritt ein, wenn ein Knoten von mehreren Kontextknoten referenziert wird. Trifft diese Bedingung zu, endet der Funktionsaufruf, da weder für den Knoten selbst, noch seine Nachfolger Regeln erstellt werden müssen. Ansonsten werden die SPF-Graphen für die linke und die rechte Seite der neuen Produktionsregel erzeugt; diese Regel bildet stets das Nichtterminalsymbol des Knotens aus dem SPF-Typ-Graphen auf den nicht-terminalen Knoten mit Knotenkardinalität *eins* ab. Anschließend wird geprüft, ob es sich um den ersten Aufruf der Funktion handelt; in diesem Fall erfolgt die Initialisierung der Graphgrammatik, indem der aktuelle SPF-Typ-Graph gesetzt wird und der soeben als linke Seite der Produktionsregel erstellte SPF-Graph als initialer Graph genutzt wird. Ist die Knotenkardinalität des gerade behandelten Knotens größer *eins*, so werden sämtliche Regeln zur Abbildung des Nichtterminalsymbols des Knotens auf alle möglichen Klonmengen in der Funktion $generateRules_{car}$ generiert. Handelt es sich bei dem Knoten um einen SPF-Knoten, also einen Blattknoten ohne Nachfolger, so ist die Summe seiner nachfolgenden Knoten *null*. In diesem Fall muss also das Nichtterminalsymbol mit der Knotenkardinalität *eins* lediglich auf das terminale Symbol, bestehend aus dem Knoten selbst und der ihm zugeordneten Attribute abgebildet werden; bereits existierende Attributwerte werden dabei übernommen. Repräsentiert der Knoten hingegen einen Kontextknoten, so muss an Hand des logischen Operators über den Umgang mit den nachfolgenden Knoten entschieden werden. Zu diesem Zweck wird in Abhängigkeit des Operators eine Funktion aufgerufen, die Produktionsregeln zur Abbildung des Nichtterminalsymbols mit Knotenkardinalität *eins* auf SPF-Graphen generiert, die sämtliche Kombinationen aus Nichtterminalsymbolen für die Nachfolgerknoten repräsentieren. Die so erzeugte Regelmenge wird mit der zuvor generierten Regelmenge vereinigt. Zum Schluss erfolgt der rekursive Aufruf der Funktion *generateRules* für jeden Nachfolger des soeben behandelten Knotens.

Der nächste Codeabschnitt zeigt den Algorithmus zur Generierung einer Menge von Regeln in Abhängigkeit der für den Knoten spezifizierten Kardinalität.

Formel 2 Algorithmus zur Erzeugung von Produktionsregeln entsprechend der Kardinalität eines Knoten

```
generateRules_CAR(GG, v, cardinality) {
    FOR (i ≔ 2; i ≤ cardinality; i + 1)
        L ≔ (N^i_v, ∅ , ∅ , N^i_v, N^i_v, ∅ , VA, VID);
        R ≔ (N^i_v ∪ N^i-1_v, ∅ , ∅ , N^i_v ∪ N^i-1_v, N^i_v ∪ N^i-1_v, ∅ , VA, VID);
        rule ≔ (v, L, R); // Regel: N^i_v → N^i_v, N^i-1_v
        GG.PR ≔ GG.PR ∪ rule; // Neue Regel hinzufügen
        R ≔ L;
        L ≔ (N_v, ∅ , ∅ , N_v, N_v, ∅ , VA, VID);
        rule ≔ (v, L, R); // Regel: N_v → N^i_v
        GG.PR ≔ GG.PR ∪ rule; // Neue Regel hinzufügen
    END-FOR
}
```

Der folgende Algorithmus demonstriert die Umsetzung der Funktion $generateRules_{AND}$, die eine Regel kreiert, deren linke Seite aus dem nicht-terminalen Knoten für die Knotenkardinali-

tät *eins* besteht und deren rechte Seite das Terminalsymbol des Knotens und dessen Attributliste umfasst, gefolgt von einer Sequenz über die Nichtterminalsymbole der nachfolgenden Knoten.

Formel 3 Algorithmus zur Erzeugung von Produktionsregeln für Knoten mit AND-Operator

```
generateRules_AND(GG, v, succNodes, A, VA) {
      v_1...v_|succNodes| := succNodes;
      L := (N^1_v, ∅ , N^1_v, N^1_v, ∅ , ∅ , VA, VID);
      R := (v∪N_v1∪...∪N_v|succNodes|,(v, N_v1)∪...∪(v, N_v|succNodes|), v, N_v1∪...∪N_v|succNodes|,
v, A, VA, VID);
      rule := (v, L, R); // Regel: N^1_v → [v, attr] (N_v1, N_v2, ..., N_v|succNodes|)
      GG.PR := GG.PR ∪ rule;
}
```

Da für Knoten mit XOR-Operator die Anzahl der notwendigen Produktionsregeln von der Menge der Nachfolger abhängt, wird nach dem Funktionsaufruf eine Schleife solange durchlaufen, bis für jeden nachfolgenden Knoten eine entsprechende Regel erzeugt wurde.

Formel 4 Algorithmus zur Erzeugung von Produktionsregeln für Knoten mit XOR-Operator

```
generateRules_XOR(GG, v, succNodes, A, VA) {
      v_1...v_|succNodes| := succNodes;
      FOR (i := 1; i ≤ |succNodes|; i + 1)
          L := (N^1_v, ∅ , ∅ , N^1_v, N^1_v, ∅ , VA, VID);
          R := (v∪N_vi,(v, N_vi), v, N_vi, v, A, VA, VID);
          rule := (v, L, R); // Regel: N^1_v → [v, attr] (N_v1)
          GG.PR := GG.PR ∪ rule;
      END-FOR
}
```

Der Algorithmus zur Generierung von Produktionsregeln für Knoten mit OR-Operator umfasst sogar zwei Schleifen. Die erste Schleife iteriert in aufsteigender Reihenfolge über die nachfolgenden Knoten; die zweite Schleife setzt den Index zunächst auf die Gesamtanzahl der Nachfolger und reduziert diesen mit jedem Iterationsschritt, bis der Index kleiner ist als der Index der ersten Schleife. Auf diese Weise lassen sich alle möglichen Kombinationen für die Auswahl der nachfolgenden Knoten kreieren.

Formel 5 Algorithmus zur Erzeugung von Produktionsregeln für Knoten mit OR-Operator

```
generateRules_OR(GG, v, succNodes, A, VA) {
      v_1...v_|succNodes| := succNodes;
      FOR (i := 1; i ≤ |succNodes|; i + 1)
          FOR (k = |succNodes|; k ≥ i; k - 1)
              L := (N^1_v, ∅ , ∅ , N^1_v, N^1_v, ∅ , VA, VID);
              R := (v∪N_vi∪N_vi+1∪...∪N_vk, (v, N_vi)∪(v, N_vi+1)∪...∪(v, N_vk), v, N_vi∪
N_vi+1∪...∪N_vk, v, A, VA, VID);
              rule := (v, L, R); // Regel: N^1_v → [v, attr] (N_vi,...N_vk)
              GG.PR := GG.PR ∪ rule;
          END-FOR
      END-FOR
}
```

Da sich der OPT-Operator nur in dem Punkt vom OR-Operator unterscheidet, dass es auch möglich ist, keinen nachfolgenden Knoten zu selektieren, kann die Realisierung so erfolgen, dass nach dem Anlegen dieser zusätzlichen Produktionsregel, die Funktion *generateRules*$_{OR}$ aufgerufen wird.

Formel 6 Algorithmus zur Erzeugung von Produktionsregeln für Knoten mit OPT-Operator

```
generateRules_OPT(GG, v, succNodes, A, VA) {
      L := (N¹_v, ∅ , ∅ ,N¹_v, N¹_v, ∅ , VA, VID);
      R := (v, ∅ , v, ∅ , v, A, VA, VID);
      rule := (v, L, R); // Regel: N¹_v → [v, attr]
      GG.Prod := GG.Prod ∪ rule;
      generateRules_OR(GG, v, succNodes, A, VA);
}
```

4.4.2.2 Initialisierung von SPF-Graphen und Anwendung von Produktionsregeln

Die Ableitung einer neuen Prozessdefinition aus dem Prozesswissen, das durch den SPF-Typ-Graphen repräsentiert wird, beginnt stets mit der Initialisierung des SPF-Graphen. Zu diesem Zweck wird der SPF-Graph auf das Startsymbol der Graphgrammatik gesetzt. Der Funktion *initSPFGraph* muss also lediglich die Graphgrammatik des SPF-Typ-Graphen übergeben werden, für den ein neuer SPF-Graph erzeugt werden soll.

Formel 7 Initialisierung von SPF-Graphen

```
initSPFGraph(GG) {
      SG := START^GG;
}
```

Anschließend kann die Auswahl und Anwendung von Produktionsregeln auf Knoten des SPF-Graphen erfolgen. Die konkrete Realisierung der Produktionsregeln hängt von der Art der Implementierung des SPF-Graphen ab. Wird der SPF-Graph z. B. als Adjazenzliste umgesetzt, so müssen die Regeln Manipulationsoperationen auf dieser Liste entsprechen. Gemäß der jeweiligen Realisierung erfolgt die Anwendung einer Regel durch den Aufruf der Funktion *applyRule(GG, SG,* v^{SG}_{NT}*, rule)*, wobei es sich bei *GG* um die Graphgrammatik handelt, *SG* dem zu manipulierenden SPF-Graphen entspricht, v^{SG}_{NT} der zu ersetzende nicht-terminale Knoten im SPF-Graphen ist und *rule* eine Regel aus der Graphgrammatik, die sich auf diesen Knoten bezieht. Es gilt zu beachten, dass v^{SG}_{NT} an Hand seiner Knotenidentität im SPF-Graphen eindeutig identifiziert werden kann.

Formel 8 Algorithmus zur Transformation von SPF-Graphen durch Regelanwendung

```
applyRule(GG, SG, v^SG_NT, rule) {
      //Prüfen der Anwendbarkeit der Regel
      IF (rule ∈ PR^GG ∧ v^SG_NT ∈ o(L^rule))
         //Vorgängerknoten bestimmen
         pred-node^SG := source(v^SG_NT);
      END IF
      IF (pred-node^SG ≠ ∅ )
```

```
        //Kante zwischen SG und v^SG_NT entfernen;
        E^SG ≔ E^SG - (pred-node^SG, v^SG_NT);
    }
    //v^SG_NT aus SG entfernen
    V^SG ≔ V^SG - v^SG_NT;
    //R in SG einfügen
    SG ≔ SG ∪ R^rule;
    IF (pred-node^SG ≠ ∅ )
        //Neu eingefügten Knoten feststellen
        R ≔ {R ∈ o(R^rule)| source(R^R) = ∅ };
        //Fehlende Kanten ergänzen
        E^SG ≔ E^SG ∪ (pred-node^SG, R^R);
    END IF
}
```

Die Funktion überprüft zunächst die Anwendbarkeit der Regel auf den übergebenen nicht-terminalen Knoten. Grundlegende Bedingungen sind, dass die Regel Bestandteil der Graphgrammatik ist und dass es sich bei v^{SG}_{NT} um ein Abbild der linken Seite der Produktionsregeln im SPF-Graphen handelt. Sind diese Voraussetzungen erfüllt, muss als nächstes der direkte Vorgänger des zu ersetzenden Knoten ermittelt werden; außer beim nicht-terminalen Startsymbol der Graphgrammatik ist *pred-node*^{SG} immer ungleich der leeren Menge. In diesem Fall wird die Kante, die *pred-node*^{SG} mit v^{SG}_{NT} verbindet aus dem SPF-Graphen entfernt, bevor der nicht-terminale Knoten selbst gelöscht wird. Anschließend wird die rechte Seite der Produktionsregeln in den SPF-Graphen eingefügt. Ist *pred-node*^{SG} nicht leer, wird der Wurzelknoten des neu eingefügten Abbilds der rechten Seite anschließend mit dem ehemaligen Vorgängerknoten von v^{SG}_{NT} verbunden. Damit ist die Anwendung der Produktionsregel auf v^{SG}_{NT} abgeschlossen.

4.4.3 Berücksichtigung von Constraint-Relationen bei der Konfiguration

Bei der Erstellung von Graphgrammatiken zur Ableitung von SPF-Graphen aus SPF-Typ-Graphen wurde gezeigt, wie die hierarchische Struktur des Graphen, Knotenkardinalitäten und logische Operatoren bei der Erzeugung der Regelmenge berücksichtigt werden. Es besteht jedoch auch die Möglichkeit, durch Constraint-Relationen horizontale Abhängigkeiten zwischen den Knoten von SPF-Typ-Graphen zu definieren. In diesem Kapitel soll nun beschrieben werden, auf welche Art und Weise Constraints die Entwicklung von SPF-Graphen durch Regelanwendung beeinflussen.

In der Literatur zur Feature-Modellierung wurden bereits Konzepte vorgestellt, wie die Formalisierung bzw. Umsetzung von Constraints erfolgen könnte. Einige Autoren demonstrieren, wie Constraints durch Xpath [WWWC 2005] Ausdrücke realisiert werden [Antkiewicz & Czarnecki 2004, Cechticky et al. 2004]. Auf Basis der XPath-Ausdrücke lässt sich automatisiert überprüfen, ob eine gegebene Konfiguration, den im Feature-Modell festgelegten Constraints gerecht wird; die Umsetzung erfolgt durch die Anwendung der XPath-Ausdrücke auf die XML-Repräsentation der Konfiguration. Die Nutzung der »Object Constraint Language« (OCL) [OMG 2010], um Constraints in Feature-Modellen darzustellen, wurde in ihrer alten Version 2.0 in mehreren Arbeiten untersucht [Streitferdt et al. 2003, Czarnecki & Kim 2005,

Czarnecki & Pietroszek 2006]. Im Kontext einiger Lösungen wurden SAT-Solver[42] zur Überprüfung der Erfüllbarkeit des durch die Constraints spezifizierten Regelwerks getestet [Batory 2005, Czarnecki & Kim 2005]. Die erzeugten OCL-Ausdrücke nehmen z. B. folgende Form an [Czarnecki & Kim 2005]:

Context PaymentMethod inv:
(PaymentType.CreditCard.isSelected() or
PaymentType.DebitCard.isSelected()) implies
GatewayRef.isSelected()

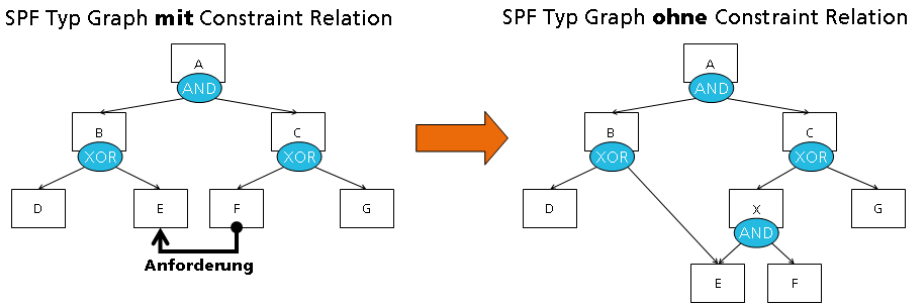
Während die meisten Lösungsansätze zur Berücksichtigung von Constraints in Feature-Modellen Klone von Features und die damit verbundenen speziellen Anforderungen nicht adressieren, beziehen Czarnecki & Kim Klone in ihre Betrachtungen mit ein. Allerdings zeigt sich an dem obigen Beispiel, dass die spezifizierten OCL-Ausdrücke Freiraum für Interpretationen gewähren. So wird nicht deutlich, ob für jeden selektierten Knoten vom Typ *CreditCard* und *DebitCard* ein eigener Knoten *GatewayRef* gesetzt werden muss oder ob ein Knoten *GatewayRef* für mehrere Knoten vom Typ *CreditCard* oder *DebitCard* ausreicht. Auch machen die Konzepte nicht deutlich, was in Fällen unternommen wird, in denen Teile der bereits vorgenommenen Konfiguration aufgrund einer erst später wirksamen gewordenen Constraint-Bedingung rückgängig gemacht und verändert werden müssen. Neben der teilweise noch unklaren Semantik besteht ein wesentlicher Nachteil der aufgeführten Methodiken darin, dass XPATH- oder OCL-Ausdrücke eigene, semi-formale Modelle zugrunde liegen, die neben dem eigentlichen Feature-Modell existieren, so dass aufwändige und fehleranfällige Modellabgleiche stattfinden müssen, um die Konsistenz zwischen Feature-Modell und Constraint Spezifikation zu gewährleisten. Beispielsweise ist die Feststellung, ob ein Constraint mit dem im Feature-Modell definierten hierarchischen Aufbau und den spezifizierten logischen Verknüpfungen harmoniert, auf Basis des oben angegebenen OCL-Ausdrucks nur schwierig zu treffen. Wegen der genannten Probleme soll in dieser Arbeit die Verwaltung einer separaten Repräsentation für Constraints vermieden werden; stattdessen werden Lösungsansätze entwickelt, wie sich Constraints durch geeignete Strukturierung des SPF-Typ-Graphen unterstützen lassen oder wie Constraints alternativ durch nachträgliche Konfiguration, Rekonfiguration oder Reduktion des SPF-Graphen durchgesetzt werden können.

4.4.3.1 Constraint verträgliche Strukturierung von SPF-Typ-Graphen

Eine Möglichkeit zur Berücksichtigung von Constraints besteht in der gezielten Strukturierung des SPF-Typ-Graphen durch Nutzung bereits vorhandener Konzepte (Hierarchie, Kardinalitäten, logische Operatoren). In Kapitel 4.3.6 wurde dieses Verfahren bereits am Beispiel von Constraint-Relationen zwischen direkten Nachfolgern desselben Kontextknotens vorgestellt. Die nachfolgende Abbildung demonstriert an einem anderen Beispiel wie ein SPF-Typ-Graph verändert werden kann, um den gewünschten Constraint durch hierarchische Beziehungen und logische Operatoren auszudrücken.

[42] Bei SAT-Solvern handelt es sich um Programme zur Lösung des aussagenlogischen Erfüllbarkeitsproblem (SAT).

Abbildung 67 Beispiel für eine Constraint verträgliche Strukturierung von SPF-Typ-Graphen

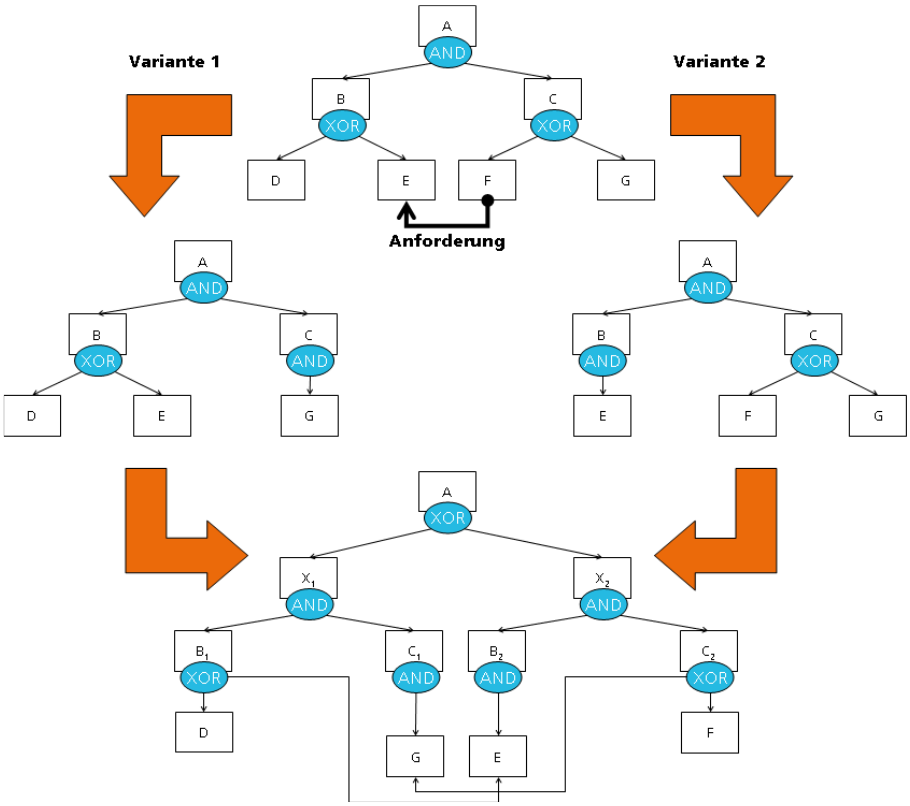


Gemäß SPF-Typ-Graphen existiert zwischen den Knoten *F* und *E* eine Anforderungsbedingung; d. h. kommt im SPF-Graphen ein *F* vor, muss der Graph mindestens auch ein *E* unterhalb des Wurzelknotens *A* umfassen. In dem Beispiel wird daher ein zusätzlicher Knoten *X* eingefügt, der als Nachfolger *E* und *F* referenziert. Diese Art der Strukturierung entspricht der einfachsten Variante zur Unterstützung von Constraint-Relationen. Wie in dem obigen Beispiel kann sie jedoch dazu führen, dass sich der Kontext, in dem sich die SPF-Knoten und die mit ihnen assoziierten Prozessfragmente befinden, ändert, was nicht in jedem Fall sinnvoll oder erwünscht ist. Denn während im ursprünglichen SPF-Typ-Graphen *E* nur unterhalb von *B* vorkommen konnte, ist *E* in der veränderten Version nun auch ein indirekter Nachfolger von *C*.

Die Problematik soll an einem Beispiel aus der Gesundheitsdomäne verdeutlicht werden: Angenommen, am post-operativen Tag findet ein Konzil statt, in das Informationen über Untersuchungen am Aufnahmetag sowie der OP-Bericht vom OP-Tag einfließen sollen. Durch eine Strukturierung wie oben vorgestellt, würden die Untersuchungen und die Erstellung des OP-Berichts dann am Tag des Konzils wiederholt werden, was natürlich nicht gewünscht ist. Darüber hinaus ändern sich durch diese Art der Strukturierung auch die Knotenkardinalitäten; während im bisherigen SPF-Typ-Graph Knoten *E* insgesamt nur einmal vorkommen konnte, kann er im veränderten SPF-Typ-Graph zweimal enthalten sein.

Eine andere Möglichkeit der Strukturierung verdeutlicht Abbildung 68. Hier wird bewusst versucht, den Kontext, indem sich die einzelnen Knoten befinden, zu erhalten.

Abbildung 68 Beispiel für eine Kontext erhaltende, Constraint verträgliche Strukturierung von SPF-Typ-Graphen



Wie aus der Abbildung hervorgeht, resultieren aus dem Constraint zwei mögliche gültige Graphvarianten. Wird in Variante 1 *G* als Nachfolger von *C* bestimmt, kann sich der Anwender bei der Konfiguration des Knoten *B* flexibel zwischen *D* und *E* entscheiden. Bei einer vorgezogenen Entscheidung für Knoten *E* in Variante 2, kommt als Nachfolger von *C* sowohl *G* als auch *F* in Frage. Variante 1 umfasst also die Knotenmengen *{A, B, C, D, G}* und *{A, B, C, E, G}*; Variante 2 ermöglicht die Knotenmengen *{A, B, C, E, F}* und *{A, B, C, E, G}*. Während also *{A, B, C, E, G}* die Schnittmenge repräsentiert, entsprechen *{A, B, C, D, G}* und *{A, B, C, E, F}* jeweils variantenspezifischen Mengen. In dem unteren Graphen werden beide Varianten unterhalb des gemeinsamen Knotens *A* als mögliche Alternativen zusammengeführt. Die Entscheidung für eine Variante während der Konfiguration wird auf diese Weise also vorgezogen. Dies birgt den Nachteil, dass sich Fachanwender schon sehr früh auf eine Variante festlegen müssen, zumal dann, wenn der Abstand zwischen dem kleinsten gemeinsamen Kontextknoten der Teilgraphen und der Constraint-Relation sehr hoch ist. Ein weiterer Nachteil dieser Methode besteht darin, dass für ehemals identische Knoten zwei unterschiedliche Repräsentationen erzeugt werden müssen; z. B. wird *B* zu B_1 und B_2. Dies ist notwendig, da der Knoten B_1 anders konfiguriert werden kann, als Knoten B_2. Dennoch kann die Egalität der Knoten nicht mehr über ihre Identität und damit auch nicht mehr über den Graphmorphismus festgestellt werden.

Zu den hier an zwei Beispielen vorgeführten Optionen existieren natürlich noch weitere alternative Lösungsansätze zur Constraint verträglichen Strukturierung von SPF-Typ-Graphen. Allen Ansätzen ist jedoch gemein, dass sie entweder den Kontext der Knoten verändern oder die Komplexität des SPF-Typ-Graphen unverhältnismäßig erhöhen und die Konfiguration erschweren. Gibt es innerhalb des Graphen mehrere Constraint-Relationen, sind die Auswirkun-

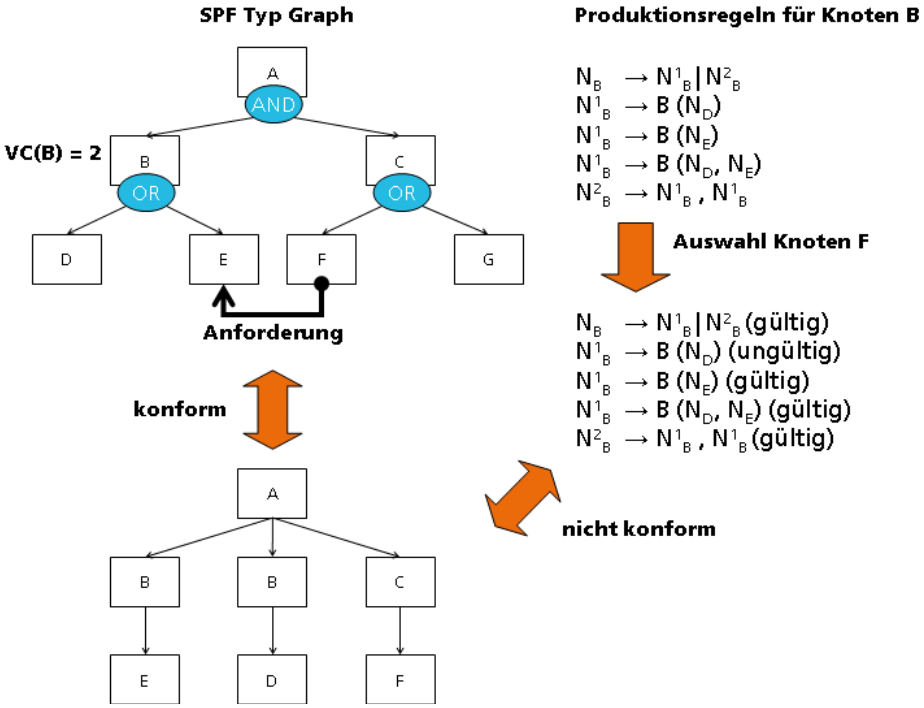
gen auf die Struktur des SPF-Typ-Graphen kaum mehr beherrschbar. Aus diesen Gründen ist das Verfahren also nur begrenzt anwendbar und es werden Algorithmen benötigt, die die Einhaltung von Constraint-Relationen bei der Konfiguration von SPF-Typ-Graphen sicherstellen.

4.4.3.2 Algorithmen zur Durchsetzung von Constraints

Kommt eine Constraint verträgliche Strukturierung des SPF-Typ-Graphen nicht in Frage, so kann die Anwendung der Regeln der Graphgrammatik nicht automatisch deren Einhaltung gewährleisten. Demnach werden also zusätzlich Algorithmen benötigt, die nach jedem Konfigurationsschritt prüfen, ob alle Constraint-Relationen im SPF-Graphen berücksichtigt sind und dies notfalls erzwingen. In diesem Kapitel werden Lösungen für die Constraints »Anforderung« und »Ausschluss« vorgestellt. Werden entsprechende Behandlungsroutinen hinzugefügt, können SPF-Typ-Graphen auch um weitere Typen von Constraints ergänzt werden, ohne Änderungen am zugrunde liegenden Konzept vornehmen zu müssen.

Der Algorithmus zur Durchsetzung von Constraints in der aktuellen Konfiguration prüft nach jeder Regelanwendung, ob eine Constraint-Relation für die terminalen Knoten des SPF-Graphen vorliegt; ist dies der Fall, wird zunächst kontrolliert, ob der Constraint erfüllt ist. Im Fall einer Anforderung bedeutet dies, dass der kleinste gemeinsame Kontextknoten der beiden Terminalknoten, zwischen denen der Constraint besteht, im SPF-Typ-Graph und im SPF-Graph identisch sein muss (siehe Kapitel 4.4.1). Der Ausschluss-Constraint bewirkt hingegen exakt die Negation dieser Bedingung. Der Ausdruck »A erfordert B« bedeutet, dass es unterhalb des kleinsten gemeinsamen Kontextknoten von *A* und *B* mindestens einen Knoten *B* geben muss, sobald während der Konfiguration ein Knoten *A* selektiert wurde. Für die Graphgrammatik hat dies die Konsequenz, dass nach der Auswahl von *A* sämtliche Regeln ungültig werden müssten, die dazu führen, dass *B* nicht unterhalb des kleinsten gemeinsamen Kontextknoten im SPF-Graphen enthalten ist (im Fall eines Ausschluss-Constraints gilt umgekehrt, dass alle Regeln, die zur Folge haben, dass *B* im SPF-Graph enthalten ist, inkorrekt sind). Eine intuitive Lösung des Problems besteht darin, alle Regeln die dazu führen, dass *B* selektiert wird als »gültig« zu markieren, während alle Regeln, die zur Folge haben, dass *B* nicht gewählt wird, auf »ungültig« gesetzt werden. Dieses Verfahren ist jedoch ausschließlich dann anwendbar, solange das Klonen von Knoten und damit auch das Klonen von Teilgraphen nicht möglich sind. Da die Markierung der Regeln als »gültig« bzw. als »ungültig« statisch ist, würden alle Klone nach demselben Muster konfiguriert werden, unabhängig davon, ob sie vom Constraint betroffen sind oder nicht. Die nachfolgende Abbildung macht diese Problematik an einem Beispiel deutlich.

Abbildung 69 Unerwünschte Einschränkungen bei der Konfiguration von Klonen



Gemäß SPF-Typ-Graph muss für einen Knoten *F* unterhalb von *A* mindestens ein Knoten *E* vorkommen. In der ursprünglichen Graphgrammatik kann alternativ zu Knoten *E* Knoten *D* gewählt werden. Sobald Knoten *F* gesetzt ist, muss die Regelmenge von Knoten *B* so verändert werden, dass sichergestellt ist, dass jede Konfiguration des SPF-Typ-Graphen mindestens ein E enthält. Da die Regel $N^1_B \rightarrow B(N_D)$ diese Bedingung nicht erfüllt, muss sie als ungültig markiert werden; die Regel steht dann nicht mehr zur Anwendung bereit. Durch diese Markierung muss nun jedoch jeder Klon von *B* so konfiguriert werden, dass er *E* umfasst, auch wenn gemäß Anforderungs-Constraint bereits ein solcher Klon ausreichen würde. Die Markierung von Produktionsregeln als gültig oder ungültig in Abhängigkeit der in der aktuellen Konfiguration enthaltenen Knoten und den mit ihnen assoziierten Constraint-Relationen greift also zu kurz und schränkt die Konfigurationsmöglichkeiten für Klone unnötig ein. Darüber hinaus müsste die Markierung der Regeln proaktiv durchgeführt werden, obwohl gar nicht sicher ist, ob der Constraint bei der Konfiguration durch den Fachanwender überhaupt verletzt werden würde.

Aus diesen Gründen wird in dieser Arbeit ein alternatives Verfahren entwickelt, das erst nach der Verletzung eines Constraints ansetzt und seine Einhaltung im SPF-Graphen nachträglich sicherstellt. Nach jeder Veränderung am SPF-Graphen (durch Auswahl und Anwendung einer Produktionsregel) wird die Funktion *checkConstraints* aufgerufen, der als Parameter der SPF-Graph die Graphgrammatik des entsprechenden SPF-Typ-Graphen übergeben werden (siehe Formel 9).

Formel 9 Algorithmus zur Überprüfung der Einhaltung der Constraint-Relationen »Anforderung« und »Ausschluss«

```
checkConstraints(GG, SG) {
      STG ≔ STG^GG;
```

```
      // Terminalknoten ohne terminale Nachfolger bestimmen
      x-nodes^SG := {v ∈ V^SG_T | ∄ u ∈ V^SG_T: v → u};
      FOR (i := 0; i < |x-nodes^SG|; i + 1)
         // Anforderungsconstraints prüfen
         y-nodes^STG := {v ∈ V^STG | ∃(g_v(x-nodes^SG[i]), v, REQUIREMENT) ∈ C^STG;
         FOR (k := 0; k < |y-nodes^STG|; k + 1)
            y-node^SG := searchNode(GG, SG, x-nodes^SG[i], y-nodes^STG[k]);
            IF (y-node^SG = ∅ )
               enforceRequirement(GG, SG, x-nodes^SG[i], y-nodes^STG[k]);
            END-IF
         END-FOR
         // Ausschlussconstraints prüfen
         y-nodes^STG := {v ∈ V^STG | (∃(x, y, EXCLUSION) ∈ C^STG: ((x = g_v(x-
nodes^SG[i]) ∧ y = v) ∨ (x = v ∧ y = g_v(x-nodes^SG[i])))};
         FOR (k := 0; k < |y-nodes^STG|; k + 1)
            y-node^SG := searchNode(GG, SG, x-nodes^SG[i], y-nodes^STG[k]);
            IF (y-node^SG ≠ ∅ )
               enforceExclusion(GG, SG, x-nodes^SG[i], y-nodes^SG);
            END-IF
         END-FOR
      END-FOR
}
```

Der Algorithmus bestimmt zunächst die terminalen Knoten der aktuellen Konfiguration, die keine Terminalknoten als Nachfolger besitzen. Anschließend wird für jeden so identifizierten Knoten geprüft, ob der Knoten über eine Constraint-Relation im SPF-Typ-Graphen mit einem anderen Knoten verbunden ist. Ist dies der Fall, so wird die Einhaltung des Constraints in der Konfiguration kontrolliert. Dies geschieht durch Aufruf der Funktion *searchNode*. Die Funktion erhält den SPF-Typ-Graphen, den SPF-Graphen, den terminalen Knoten im SPF-Graphen und den Constraint relevanten Knoten im SPF-Typ-Graphen als Parameter. *searchNode* stellt fest, ob der Constraint relevante Knoten entweder als Terminalknoten oder als Nichtterminalknoten unterhalb des kleinsten gemeinsamen Kontextknoten im SPF-Graphen enthalten ist und liefert diesen Knoten als Ergebnis zurück; konnte kein solcher Knoten identifiziert werden, so entspricht das Ergebnis der Funktion der leeren Menge. Der Algorithmus setzt dabei die Constraintbedingung aus Kapitel 4.4.1 um.

Formel 10 Algorithmus zur Identifikation von Knoten gemäß Constraintbedingung

```
searchNode(GG, SG, x-node^SG, y-node^STG) {
      STG := STG^GG;
      y-nodes^SG := {v ∈ V^SG | g_v(v) = y-node^STG ∨ (∃(y-node^STG, L, R) ∈ PR^GG: L =
(v^SG, ∅ , ∅ , v^SG, v^SG, VA^SG, VID^SG));
      FOR (i := 0; i < |y-nodes^SG|; i + 1)
         Z := {z ∈ V^SG_T | minNode(x-node^SG, y-nodes^SG[i]) →* z}; // VID der
Knoten beachten
         IF (g_v(minNode(Z)) = minNode(g_v(Z))
            // Knoten gefunden
            return y-nodes^SG[i];
         END-IF
      END-FOR
      // Keinen Knoten gefunden
      return ∅ ;
}
```

Der Algorithmus *searchNode* identifiziert zunächst alle Knoten im SPF-Graphen, deren Knotenbezeichnung mit der Bezeichnung des Constraint relevanten Knoten im SPF-Typ-Graphen

oder mit dem entsprechenden Nichtterminalsymbol übereinstimmt. Anschließend muss kontrolliert werden, ob der kleinste gemeinsame Kontextknoten des Terminalknotens *x-node*^{SG} und des Constraint relevanten Knotens *y-node*^{STG} im SPF-Graphen mit dem kleinsten gemeinsamen Kontextknoten im SPF-Typ-Graphen korrespondiert; nur dann ist die Constraintbedingung erfüllt. Um die Knoten der Menge *Z* auch im SPF-Typ-Graphen zu ermitteln, darf diese Menge natürlich nur noch terminale Knoten umfassen. Bei der Bestimmung des Kontextknoten im SPF-Graphen muss darauf geachtet werden, dass der Terminalknoten *x-node*^{SG} an Hand seiner Knotenidentität im SPF-Graphen (mittels Funktion *VID*) eindeutig identifiziert wird. Da es im SPF-Graphen auch Klone von *x-node*^{SG} geben kann, die die Constraintbedingung bereits erfüllen, würde das Ergebnis sonst verfälscht werden.

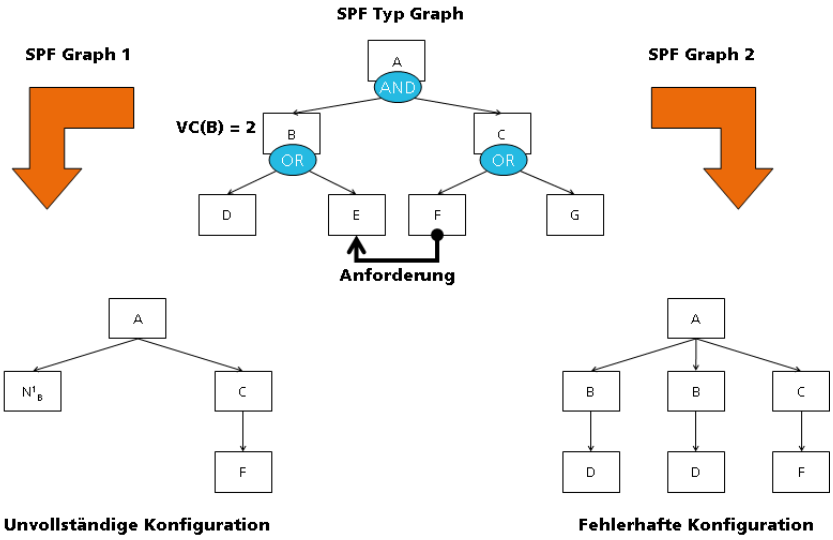
Wurde in der Funktion *checkConstraints* festgestellt, dass die aktuelle Konfiguration gegen einen Constraint verstößt, so muss nun die Einhaltung des Constraints erzwungen werden. Dies geschieht über die Funktionen *enforceRequirement* bzw. *enforceExclusion*. Während *enforceRequirement* die Knotenbezeichnung des Constraint relevanten Knoten aus dem SPF-Typ-Graphen als Parameter übergeben wird (*y-node*^{STG}), erhält *enforceExclusion* den Knoten aus dem SPF-Graphen (*y-node*^{SG}), der über die Funktion *VID* eindeutig identifiziert werden kann.

4.4.3.2.1 Durchsetzung von Anforderungs-Constraints

Bei der Sicherstellung der Erfüllung eines Anforderungs-Constraint können im Prinzip zwei mögliche Fälle voneinander abgegrenzt werden (siehe Abbildung 70):

- **Unvollständige Konfiguration:** Der Constraint relevante Knoten (bzw. sein Nichtterminalsymbol) ist in der Konfiguration nicht enthalten, da er selbst oder/und seine Vorgänger noch nicht konfiguriert wurden.
- **Fehlerhafte Konfiguration:** Die Vorgänger des Constraint relevanten Knoten wurden so konfiguriert, dass der Knoten (bzw. sein Nichtterminalsymbol) nicht in der Konfiguration enthalten ist.

Abbildung 70 Unvollständige und fehlerhafte Konfiguration in Zusammenhang mit dem Constraint »Anforderung«



Die obige Abbildung zeigt zwei mögliche Konfigurationen desselben SPF-Typ-Graphen. Für beide SPF-Graphen gilt, dass sie gegen den Anforderungs-Constraint »*F* erfordert *E*« verstoßen. Im ersten Fall wurde der Knoten N_B noch nicht konfiguriert, so dass der Knoten *E* (bzw. N_E oder N^1_E) im Graphen fehlt. Beim zweiten SPF-Graphen wurden beide Klone von *B* so konfiguriert, dass *E* deselektiert wurde. Während also bei *SPF-Graph 1* die Konfiguration derart vervollständigt werden muss, dass N_E im Graphen enthalten ist, ist bei *SPF-Graph 2* die Rekonfiguration eines Klons von *B* erforderlich.

Die Funktion *enforceRequirement* nimmt als Parameter die Graphgrammatik, den SPF-Graphen, den terminalen Knoten und die Bezeichnung des Constraint relevanten Knoten im SPF-Typ-Graphen entgegen. Als erstes wird der kleinste minimale Kontextknoten ermittelt, der sowohl im SPF-Typ-Graphen als auch im SPF-Graphen vorhanden ist; dabei müssen eventuell vorhandene Mehrfachreferenzen im SPF-Typ-Graphen beachtet werden. Anschließend bestimmt die Funktion als Konfigurationsmenge alle Knoten, die zwischen diesem minimalen Kontextknoten und dem Constraint relevanten Knoten *y-node*^{STG} liegen; die Menge umfasst außerdem den Kontextknoten und den Constraint relevanten Knoten selbst. Innerhalb dieser Menge sollen nun Kandidaten für die nachträgliche Konfiguration oder die Rekonfiguration des SPF-Graphen identifiziert werden. Die nachfolgende Programmsequenz repräsentiert den Algorithmus *enforceRequirement* in Pseudocode.

Formel 11 Hauptalgorithmus zur Durchsetzung von Anforderungs-Constraints

```
enforceRequirement(GG, SG, x-node^SG, y-node^STG) {
      STG ≔ STG^GG;
      //Bestimme kleinsten gemeinsamen Kontextknoten, der sowohl im SPF-Typ-
Graphen als auch SPF-Graphen vorhanden ist
      min-node^STG ≔ ∅ ;
      test-node^STG ≔ minNode(g_v(x-node^SG), y-node^STG);
      WHILE (min-node^STG = ∅ )
         IF (∃v ∈ V^SG: g_v(v) = test-node^STG ∧ v →* x-node^SG) // VID von x-node^SG
beachten
            min-node_STG ≔ test-node_STG;
         END-IF
         ELSE
            test-nodes^STG ≔ {v ∈ V^STG|v → test-node^STG};
            IF (|test-nodes^STG| > 1) // Bei Mehrfachreferenz
               test-node^STG ≔ minNode(test-nodes^STG);
            END-IF
            ELSE
               test-node^STG ≔ test-nodes^STG;
            END-ELSE
         END-ELSE
      END-WHILE

      //Bestimme (Re-)Konfigurationsmenge
      config-set^STG ≔ min-node^STG ∪ y-node^STG ∪ {v ∈ V^STG|min-node^STG →* v ∨ v
→* y-node^STG};
      config-nodes^SG ≔ ∅ ;
      reconfig-nodes^SG ≔ ∅ ;
      old-nodes^STG ≔ y-node^STG;
      WHILE (|old-nodes^STG| > 1 ∨ min-node^STG ∉ old-nodes^STG)
         new-nodes^STG ≔ {v ∈ config-set^STG|target(v) ∈ old-nodes^STG ∧ v ∉ new-
nodes^STG};
         FOR (i ≔ 0; i < |new-nodes_STG|; i + 1)
            new-node^SG ≔ searchNode(GG, SG, x-node^SG, new-nodes^STG[i]);
            IF (new-node^SG ∈ V^SG_NT)
               //Für Konfiguration merken
```

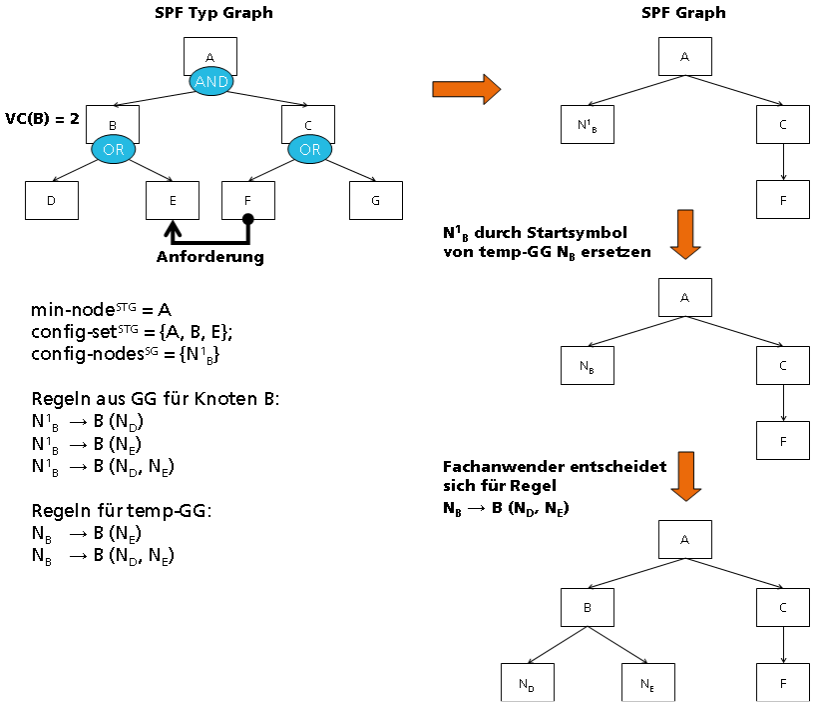
```
            config-nodes^SG := config-nodes^SG ∪ new-node^SG;
         END-IF
         ELSE IF (new-node^SG ∈ V^SG_T)
            //Für Rekonfiguration vormerken
            reconfig-nodes^SG := reconfig-nodes^SG ∪ new-node^SG;
         END-ELSE-IF
      END-FOR
      old-nodes^STG := new-nodes^STG;
   END-WHILE
   //Nachträgliche Konfiguration des SPF-Typ-Graphen
   IF (config-nodes^SG ≠ ∅ )
      config-node^SG := config-nodes^SG[0];
      IF (|config-nodes^SG| > 1)
         //Entscheidung des Fachanwenders für einen Knoten
         config-node^SG := promptUser(config-nodes^SG);
      END-IF
      enforceRequirementByConfiguration(GG, SG, config-node^SG, config-
set^STG, x-node^SG, y-node^STG);
   END-IF
   // Rekonfiguration des SPF-Typ-Graphen
   ELSE IF (reconfig-nodes^SG ≠ ∅ )
      // Unnötige Kontextknoten aus der Rekonfigurationsmenge entfernen
      reconfig-nodes^SG := reconfig-nodes^SG - {v ∈ reconfig-nodes^SG|∃u ∈
reconfig-nodes^SG: v →* u};
      reconfig-node^SG := reconfig-nodes^SG[0];
      IF(|reconfig-nodes^SG| > 1)
         // Entscheidung des Fachanwenders für einen Knoten
         reconfig-node^SG := promptUser(reconfig-nodes^SG);
      END-IF
      enforceRequirementByReconfiguration(GG, SG, reconfig-node^SG, config-
set^STG, x-node^SG, y-node^STG);
   END-ELSE-IF
}
```

Das Vorliegen von Mehrfachreferenzen im SPF-Typ-Graphen kann die Bestimmung des kleinsten gemeinsamen Kontextknoten erschweren. So ist es möglich, dass der Kontextknoten im SPF-Graphen gar nicht vorhanden ist, weil *x-node*^{SG} auf einem anderen Weg konfiguriert wurde. In diesem Fall wird geprüft, ob der Vorgänger des Kontextknoten im SPF-Graphen enthalten ist. Hat der Kontextknoten jedoch mehr als zwei Vorgänger, dann soll *min-node*^{STG} dem kleinsten gemeinsamen Kontextknoten der Vorgängerknoten entsprechen. Ausgehend von *min-node*^{STG} und *y-node*^{STG} kann nun die Konfigurationsmenge *config-set*^{STG} festgelegt werden. Innerhalb dieser Menge wird nach Knoten gesucht, die eine terminale oder nicht-terminale Entsprechung im SPF-Graphen besitzen. Die Suche beginnt bei den Vorgängerknoten *von y-node*^{STG} und setzt sich bis zu dem Wurzelknoten der Konfigurationsmenge, *min-node*^{STG}, fort.

Kann innerhalb des SPF-Graphen ein nicht-terminaler Knoten erkannt werden, der mit einem Knoten der Konfigurationsmenge übereinstimmt, so besteht die Möglichkeit der nachträglichen Konfiguration. In diesem Fall wird eine vorläufige Graphgrammatik erstellt, die auf der ursprünglichen Grammatik basiert und nur solche Regeln enthält, deren Anwendung dazu führt, dass der Constraint relevante Knoten unterhalb des kleinsten gemeinsamen Kontextknoten im SPF-Graphen enthalten ist. Der nicht-terminale Knoten im SPF-Graphen, der konfiguriert werden soll, wird durch das Startsymbol der vorläufigen Graphgrammatik ersetzt. Hierbei sei angemerkt, dass die nachträgliche Konfiguration auf dem schnellsten Weg dazu führen soll, dass der Constraint erfüllt ist; aus diesem Grund werden Klone bei der Konfiguration nicht berücksichtigt. Diese können jedoch mit Hilfe der Rekonfiguration des SPF-Graphen nachträglich hinzugefügt werden (siehe Kapitel 4.6). Existieren mehrere Möglichkeiten der Konfiguration,

so ist die Interaktion mit dem Fachanwender erforderlich. Je nach Implementierung kann die normale Konfiguration erst dann wieder aufgenommen werden, sobald der SPF-Graph das Nichtterminalsymbol des Constraint relevanten Knotens enthält; alternativ kann das System jedoch auch registrieren, dass dieser Knoten noch konfiguriert werden muss und den Anwender spätestens dann auf diese Notwendigkeit hinweisen, wenn er den SPF-Graphen in eine Prozessdefinition transformieren möchte. Abbildung 71 verdeutlicht die Funktionsweise des Algorithmus an einem Beispiel.

Abbildung 71 Funktionsweise des Algorithmus zur nachträglichen Konfiguration eines SPF-Typ-Graphen



Der nachfolgende Codeausschnitt repräsentiert den Algorithmus für die nachträgliche Konfiguration des SPF-Typ-Graphen zur Durchsetzung von Anforderungs-Constraints. Die Funktion zur Regelanwendung *applyRule* wurde bereits in Kapitel 4.4.2.2 spezifiziert. Als *conditions* Parameter wird ihr die Identität des zu konfigurierenden Knotens im SPF-Graph übergeben.

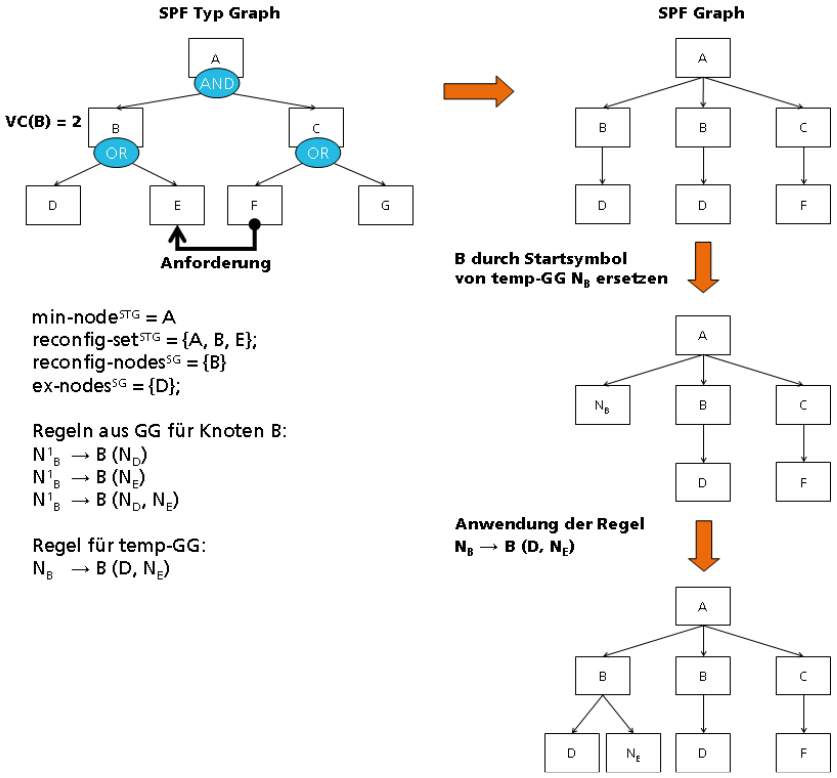
Formel 12 Algorithmus zur nachträglichen Konfiguration des SPF-Typ-Graphen zur Durchsetzung von Anforderungs-Constraints

```
enforceRequirementByConfiguration(GG, SG, config-node^SG, config-set^STG, x-node^SG,
y-node^STG) {
      //Vorläufige Graphgrammatik erstellen
      temp-GG ≔ (STG^temp-GG, PR^temp-GG, START^temp-GG);
      STG^temp-GG ≔ STG^GG;
      //Knotenbezeichnung des Terminalsymbols ermitteln
      rules ≔ {(v, L, R) ∈ PR^GG|V^L = config-node^SG};
      config-node^STG ≔ rules[0].v;
      c-nodes^STG ≔ config-node^STG;
      config-set^STG ≔ config-node^STG ∪ {v ∈ config-set^STG|config-node^STG →* v};
      WHILE (|c-nodes^STG| > 1 ∨ y-node^STG ∉ c-nodes^STG)
         succ-nodes^STG ≔ {v ∈ config-set^STG|source(v) ∈ c-nodes^STG};
```

```
        FOR (i ≔ 0; i < |c-nodes^STG|; i + 1)
           c-node^STG ≔ c-nodes^STG[i];
           old-L ≔ (N^1_c-nodeSTG, ∅ , ∅ , N^1_c-nodeSTG, N^1_c-nodeSTG, ∅ , VA, VID);
           new-L ≔ (N_c-nodeSTG, ∅ , ∅ , N_c-nodeSTG, N_c-nodeSTG, ∅ , VA, VID);
           //Relevante Regeln identifizieren
           c-rules ≔ {(v, L, R) ∈ PR^GG|L = old-L ∧(∃v ∈ V^R_NT ∃(u, L_x, R_x) ∈
PR^GG: u ∈ succ-nodes^STG};
           FOR (k ≔ 0; k < |c-rules|; k + 1)
              //Regel mit neuer linken Seite erzeugen
              new-rule ≔ (c-node^STG, new-L, c-rules[k].R);
              PR^temp-GG ≔ PR^temp-GG ∪ new-rule;
           END-FOR
           IF (c-node^STG ≔ config-node^STG ∧ config-node^SG ≠ N_c-nodeSTG)
              //Einzelnen Knoten im SPF-Graphen ersetzen; VID beachten
              Nbh ≔ {v ∈ V^SG|v → config-node^SG};
              E^SG ≔ E^SG - {(x, config-node^SG) ∈ E^SG|x ∈ Nbh};
              V^SG ≔ V^SG - config-node^SG;
              V^SG ≔ V^SG ∪ N_c-nodeSTG;
              E^SG ≔ E^SG ∪ {(x, N_c-nodeSTG)|x ∈ Nbh};
              START^temp-GG ≔ new-L;
              config-nodes^SG ≔ searchNode(GG, SG, x-node^SG, N_c-nodeSTG);
           END-IF
        END-FOR
        c-nodes^STG ≔ succ-nodes^STG;
     END-WHILE
     //SPF-Graph mit Hilfe der vorläufigen Graphgrammatik konfigurieren
     WHILE (searchNode(GG, SG, x-node^SG, y-node^STG) = ∅ ∧ config-nodes^SG ≠ ∅ )
        config-node^SG ≔ config-nodes^SG[0];
        IF(config-nodes^SG > 1)
           // Entscheidung des Fachanwenders für einen Knoten
           config-node^SG ≔ promptUser(config-nodes^SG);
        END-IF
        pred-nodes^SG ≔ {v ∈ V^SG_T|v → config-node^SG} // Vorgänger merken
        rules ≔ {(v, L, R) ∈ PR^temp-GG|V^L = config-node^SG};
        rule ≔ rules[0];
        IF(rules > 1)
           //Entscheidung des Fachanweders für eine Regel
           rule ≔ promptUser(rules);
        END-IF
        applyRule(temp-GG, SG, config-node^SG, rule);
        t-config-nodes^SG ≔ {v ∈ V^SG_T|source(v) ∈ pred-nodes^SG};
        config-nodes^SG ≔ {v ∈ V^SG_NT|source(v) ∈ t-config-nodes^SG ∧ (∃ (u, L,
R) ∈ PR^temp-GG: V^L = v)}; // Nächste zu konfigurierende Knoten bestimmen
     END-WHILE
}
```

Ist kein nicht-terminaler Knoten im SPF-Graph vorhanden, der die nachträgliche Konfiguration des benötigten Teilgraphen erlaubt hätte, so muss stattdessen ein im SPF-Graph bereits existierender terminaler Knoten rekonfiguriert werden. Der Algorithmus für *enforceRequirement-ByReconfiguration* ähnelt in großen Teilen dem Algorithmus für *enforceRequirementByConfiguration*. Gibt es die Möglichkeit, den bestehenden Teilgraphen unterhalb des neu zu konfigurierenden Knotens zu erhalten, so werden die entsprechenden Knoten allerdings bei der Bildung der rechten Seite der neuen Regel berücksichtigt. Bei der Identifikation der relevanten Regeln in der originären Graphgrammatik muss sichergestellt werden, dass tatsächlich alle nicht-terminalen Knoten auf der rechten Seite eine terminale Entsprechung in der Rekonfigurationsmenge besitzen; sonst bestünde die Gefahr, dass bei der Rekonfiguration mehr Klone hinzugefügt werden können, als im SPF-Typ-Graphen vorgesehen ist. Die folgende Abbildung verdeutlicht die Funktionsweise des Algorithmus beispielhaft.

Abbildung 72 Funktionsweise des Algorithmus zur Rekonfiguration eines SPF-Typ-Graphen



Die anschließende Programmsequenz repräsentiert den Algorithmus der Funktion *enforceRequirementByReconfiguration* in Pseudocode.

Formel 13 Algorithmus zur Rekonfiguration des SPF-Typ-Graphen zur Durchsetzung von Anforderungsconstraints

```
enforceRequirementByReconfiguration(GG, SG, reconfig-node^SG, reconfig-set^STG, x-node^SG, y-node^STG) {
      STG = STG^GG;
      //Vorläufige Graphgrammatik erstellen
      temp-GG ≔ (STG^temp-GG, PR^temp-GG, START^temp-GG);
      STG^temp-GG ≔ STG;
      c-nodes^STG ≔ g_v(reconfig-node^SG);
      reconfig-set^STG ≔ g_v(reconfig-node^SG) ∪ {v ∈ reconfig-set^STG|g_v(reconfig-node^SG) →* v};
      ex-nodes^SG ≔ ∅ ;
      IF (VO^STG(g_v(reconfig-node^SG)) ≠ XOR)
         ex-nodes^SG ≔ {v ∈ V^SG|source(v) = reconfig-node^SG ∧ v ∉ reconfig-set^STG};
      END-IF
      WHILE (|c-nodes^STG| > 1 ∨ y-node^STG ∉ c-nodes^STG)
         succ-nodes^STG ≔ {v ∈ reconfig-set^STG|source(v) ∈ c-nodes^STG};
         FOR (i ≔ 0; i < |c-nodes^STG|; i + 1)
            c-node^STG ≔ c-nodes^STG[i];
            old-L ≔ (N¹_c-nodeSTG, ∅ , ∅ , N¹_c-nodeSTG, N¹_c-nodeSTG, ∅ , VA, VID);
            new-L ≔ (N_c-nodeSTG, ∅ , ∅ , N_c-nodeSTG, N_c-nodeSTG,∅ , VA, VID);
            //Relevante Regeln identifizieren
            c-rules ≔ {(v, L, R) ∈ PR^GG|L = old-L ∧ (∀v ∈ V^R_NT ∃(u, L_x, R_x) ∈ PR^GG: u ∈ succ-nodes^STG};
            FOR (k ≔ 0; k < |c-rules|; k + 1)
```

```
            IF (c-node^STG = reconfig-node^STG ∧ ex-nodes^SG ≠ ∅ )
               // Regel erzeugen, deren rechte Seite die existierende
Struktur erhält
               new-R ≔ c-rules[k].R;
               V^new-R ≔ V^new-R ∪ ex-nodes_SG;
               E^new-R ≔ E^new-R ∪ {(x, y) ∈ E^SG|x, y ∈ ex-nodes^SG ∨ (x = r^new-R ∧ y
∈ ex-nodes^S E^new-R ^G)};
               new-rule ≔ (c-node^STG, new-L, new-R);
            END-IF
            ELSE
               new-rule ≔ (c-node^STG, new-L, c-rules[k].R);
            END-ELSE
            PR^temp-GG ≔ PR^temp-GG ∪ new-rule;
         END-FOR
         IF (c-node^STG = reconfig-node^STG ∧ reconfig-node^SG ≠ N_c-nodeSTG)
            //Teilgraph durch Nichtterminalknoten ersetzen
            Nbh ≔ {v ∈ V^SG|v → reconfig-node^SG};
            E^SG ≔ E^SG - {(x, y) ∈ E^SG|(x ∈ Nbh ∧ y = reconfig-node^SG) ∨ (x =
reconfig-node^SG ∧ y ∈ ex-nodes^SG) ∨ (x, y ∈ ex-nodes^SG)};
            V^SG ≔ V^SG - reconfig-node^SG - ex-nodes^SG;
            V^SG ≔ V^SG ∪ N_c-nodeSTG;
            E^SG ≔ E^SG ∪ {(x, N_c-nodeSTG)|x ∈ Nbh};
            START^temp-GG ≔ new-L;
            reconfig-nodes^SG ≔ searchNode(GG, SG, x-node^SG, N_c-nodeSTG);
         END-IF
      END-FOR
      c-nodes^STG ≔ succ-nodes^STG;
   END-WHILE
   //SPF-Graph mit Hilfe der vorläufigen Graphgrammatik konfigurieren
   WHILE (searchNode(GG, SG, x-node^SG, y-node^STG) = ∅ ∧ reconfig-nodes^SG ≠ ∅
)
      reconfig-node^SG ≔ reconfig-nodes^SG[0];
      IF(reconfig-nodes^SG > 1)
         // Entscheidung des Fachanwenders für einen Knoten
         reconfig-node^SG ≔ promptUser(config-nodes^SG);
      END-IF
      pred-nodes^SG ≔ {v ∈ V^SG_T|v → reconfig-node^SG} // Vorgänger merken
      rules ≔ {(v, L, R) ∈ PR^temp-GG|V^L = reconfig-node^SG};
      rule ≔ rules[0];
      IF(rules > 1)
         // Entscheidung des Fachanweders für eine Regel
         rule ≔ promptUser(rules);
      END-IF
      applyRule(temp-GG, SG, reconfig-node^SG, rule);
      t-reconfig-nodes^SG ≔ {v ∈ V^SG_T|source(v) ∈ pred-nodes^SG};
      reconfig-nodes^SG ≔ {v ∈ V^SG_NT|source(v) ∈ t-reconfig-nodes^SG ∧ (∃(u,
L, R) ∈ PR^temp-GG: V^L = v)}; // Nächste zu konfigurierende Knoten bestimmen
   END-WHILE
}
```

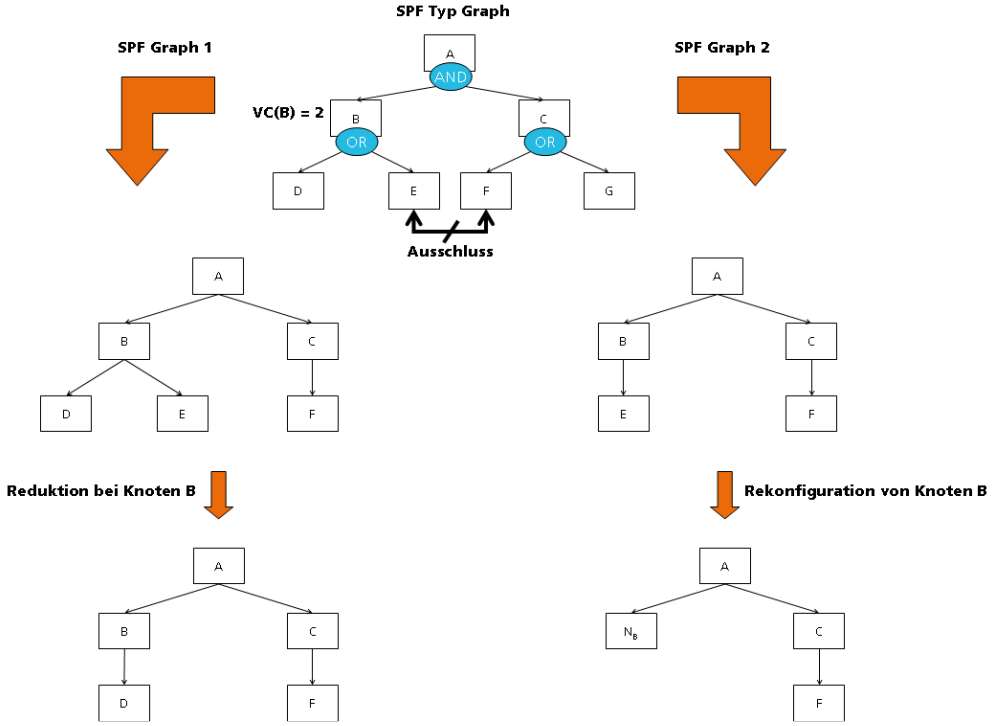
4.4.3.2.2 Durchsetzung von Ausschluss-Constraints

Im Gegensatz zu Anforderungs-Constraints geht die Verletzung eines Ausschluss-Constraints immer auf eine fehlerhafte Konfiguration zurück. In Abhängigkeit der aktuellen Konfiguration kommen zur Auflösung des Konflikts zwei Möglichkeiten in Frage (siehe Abbildung 73):

- **Reduktion:** Bei der Reduktion wird der Teilgraph, der den Constraint relevanten Knoten enthält, aus dem SPF-Graphen entfernt, wobei alle Knoten, des Teilgraphen, die nicht in Zusammenhang mit dem Konflikt stehen, erhalten bleiben. Da dieses Vorgehen keinen zusätzlichen Aufwand für den Fachanwender bedeutet, handelt es sich hierbei um die Strategie der Wahl.

– **Rekonfiguration:** Lassen die Festlegungen im SPF-Typ-Graphen eine Reduktion nicht zu (z. B. weil mindestens ein Knoten gesetzt sein muss), muss der Teilgraph, der den Constraint relevanten Knoten enthält, ab einem geeignetem Kontextknoten aus so rekonfiguriert werden, dass der Knoten, der den Konflikt auslöst, nicht mehr im SPF-Graph enthalten ist.

Abbildung 73 Reduktion oder Rekonfiguration zur Durchsetzung von Ausschluss-Constraints in SPF-Graphen



In dem obigen Beispiel wurde Knoten *B* im *SPF-Graph 1* so konfiguriert, dass ihm Knoten *D* und *E* als Nachfolger zugeordnet sind. Es ist also möglich, den SPF-Graph ab Knoten *B* derart zu reduzieren, dass Knoten *E* nicht mehr im Graph vorkommt; Knoten *D* bleibt hingegen erhalten und *B* ist weiterhin korrekt konfiguriert. Bei *SPF-Graph 2* wurde als Nachfolger von *B* lediglich Knoten *E* ausgewählt. Weder Knoten *B* noch Knoten *A* lassen sich so reduzieren, dass die Vorgaben bezüglich der logischen Verknüpfungen im SPF-Typ-Graphen nicht verletzt werden. Aus diesem Grund muss also Knoten *B* rekonfiguriert werden. Der Algorithmus für die Funktion *enforceExclusion* dient also der Ermittlung von Knoten, die eine Reduktion des SPF-Graphen oder eine Rekonfiguration des SPF-Typ-Graphen ermöglichen und gleichzeitig auch der Durchführung der notwendigen Maßnahmen zur Durchsetzung des Constraints.

Formel 14 Algorithmus zur Durchsetzung von Ausschluss-Constraints

```
enforceExclusion(GG, SG, x-node^SG, y-node^SG) {
      STG ≔ STG^GG;
      //Bestimme kleinsten gemeinsamen Kontextknoten im SPF-Graphen
      min-node^SG ≔ minNode(x-node^SG, y-node^SG);
      //Bestimme Reduktionsmenge
      reduction-set^SG ≔ min-node^SG ∪ y-node^SG ∪ {v ∈ V^SG|min-node^SG →* v ∨ v
→* y-node^SG};
```

```
      reduction-nodes^SG := ∅ ;
      old-nodes^SG := y-node^SG;
      WHILE (|old-nodes^STG| > 1 ∨ min-node^SG ∉ old-nodes^SG)
         new-nodes^SG := {v ∈ reduction-set^SG|target(v) ∈ old-nodes^SG ∧ v ∉
new-nodes^SG};
         FOR (i := 0; i < |new-nodes^SG|; i + 1)
            new-node^SG := new-nodes^SG[i];
            succ-nodes^SG := {v ∈ V^SG|new-node^SG → v};
            IF (VO^STG(g_v(new-node^SG)) ≠ AND ∧ (∃ u ∈ succ-nodes^SG: u ∉
reduction-set^SG ∨ VO^STG(g_v(new-node^SG)) = OPT))
               //Für Reduktion vormerken
               reduction-nodes^SG := reduction-nodes^SG ∪ new-node^SG;
            END-IF
            ELSE IF(VO^STG(gv(new-node^SG)) ≠ AND ∧ (∃ u ∈ V^STG: g_v(new-node^SG) →
u ∧ ¬(∃ z ∈ reduction-set^SG: g_v(z) = u)))
               //Für Rekonfiguration vormerken
               reconfig-nodes^SG := reconfig-nodes^SG ∪ new-node^SG;
            END-ELSE-IF
         END-FOR
         old-nodes^SG := new-nodes^SG;
      END-WHILE

      // Reduktion des SPF-Graphen
      IF (reduction-nodes^SG ≠ ∅ )
         //Unnötige Kontextknoten aus der Reduktionsmenge entfernen
         reduction-nodes^SG := reduction-nodes^SG - {v ∈ reduction-nodes^SG|∃u ∈
reduction-nodes^SG: v →* u};
         reduction-node^SG := reduction-nodes^SG[0];
         IF(|reduction-nodes^SG| > 1)
            //Entscheidung des Fachanwenders für einen Knoten
            reduction-node^SG := promptUser(reduction-nodes^SG);
         END-IF
         //Konflikt-auslösenden Teilgraph aus SG entfernen
         succ-nodes^SG := {v ∈ reduction-set^SG|reduction-node^SG → v};
         c-node-set^SG := succ-nodes^SG ∪ {v ∈ V^SG|∃ u ∈ succ-nodes^SG: u →* v};
         E^SG := E^SG - {(x, y) ∈ E^SG|(x = reduction-node^SG ∨ x ∈ c-node-set^SG) ∧
y ∈ c-node-set^SG};
         V^SG := V^SG - c-node-set^SG;
      END-IF
      //Rekonfiguration des SPF-Typ-Graphen
      ELSE-IF (reconfig-nodes^SG ≠ ∅ )
         // Unnötige Kontextknoten aus der Rekonfigurationsmenge entfernen
         reconfig-nodes^SG := reconfig-nodes^SG - {v ∈ reconfig-nodes^SG|∃u ∈
reconfig-nodes^SG: v →* u};
         reconfig-node^SG := reconfig-nodes^SG[0];
         IF(|reconfig-nodes^SG| > 1)
            // Entscheidung des Fachanwenders für einen Knoten
            reconfig-node^SG := promptUser(reconfig-nodes^SG);
         END-IF
         //Konfliktauslösenden Teilgraph durch Nichtterminalknoten ersetzen
         Nbh = {v ∈ V^SG|v → reconfig-node^SG};
         c-node-set^SG := reconfig-node^SG ∪ {v ∈ V^SG|reconfig-node^SG →* v};
         E^SG := E^SG - {(x, y) ∈ E^SG|(x ∈ Nbh ∨ x ∈ c-node-set^SG) ∧ y ∈ c-
node-set^SG};
         V^SG := V^SG - c-node-set_SG;
         V^SG = V^SG ∪ N_reconfig-nodeSG;
         E^SG = E^SG ∪ {(x, N_reconfig-nodeSG)|x ∈ Nbh};
      END-ELSE-IF
      ELSE
         //Konfliktknoten kann nicht entfernt werden
         enforceExclusion(GG, SG, y-node^SG, x-node^SG);
      END-ELSE
}
```

Da sowohl *x-node*^{SG} als auch *y-node*^{SG} Bestandteil des SPF-Graphen sind, kann der kleinste gemeinsame Kontextknoten viel einfacher als in *enforceRequirement* ermittelt werden, da kein Vergleich mit dem Kontextknoten im SPF-Typ-Graphen notwendig ist. Anschließend wird geprüft, welche Knoten sich für eine Reduktion bzw. für eine Rekonfiguration eignen. Existieren Knoten, die für eine Reduktion des SPF-Graphen in Frage kommen, so wird der Teilgraph, der den Konflikt verursacht, einfach aus dem SPF-Graphen entfernt. Gibt es die Möglichkeit der Reduktion nicht, so wird geprüft, ob eine Rekonfiguration durchgeführt werden kann; in diesem Fall wird der Teilgraph ab dem identifizierten Rekonfigurationsknoten aus dem SPF-Graph entfernt und stattdessen durch den Nichtterminalknoten des Rekonfigurationsknotens ersetzt. Anschließend kann der Fachanwender die Konfiguration des Knoten erneut durchführen. Dabei kann jedoch nicht automatisch ausgeschlossen werden, dass der Konflikt auslösende Knoten wieder in die Konfiguration mit eingeschlossen wird; ist dies der Fall müsste der Algorithmus erneut ausgeführt werden, wobei die Rollen von *x-node*^{SG} und *y-node*^{SG} vertauscht sind. Dies ist auch notwendig, wenn sich *y-node*^{SG} nicht aus dem SPF-Graphen entfernen lässt. Gemäß den Korrektheitskriterien für SPF-Typ-Graphen aus Kapitel 4.3.6 muss es aber möglich sein, mindestens einen der Knoten *x-node*^{SG} oder *y-node*^{SG} aus dem SPF-Graphen zu löschen.

Sollen SPF-Typ-Graphen um zusätzliche, benutzerdefinierte Constrainttypen ergänzt werden, so muss zunächst das Metamodell für SPF-Typ-Graphen aus Kapitel 4.3.5 entsprechend erweitert werden. Nach der exakten Klärung der Semantik des neuen Constrainttyps, steht fest, ob der Constraint Einfluss auf den Aufbau von SPF-Graphen und/oder Prozessdefinitionen hat. Gibt es Auswirkungen während der Konfigurations- bzw. Rekonfigurationsphase, so muss die Funktion *checkConstraints* um eine Routine zur Prüfung der Einhaltung des Constraints angereichert werden; außerdem ist es notwendig eine entsprechende Funktion *enforceX* zu erstellen, wobei *X* der Bezeichnung des neuen Constraints entspricht. Determiniert der Constraint die Struktur von Prozessdefinitionen, so muss die Transformation unter Beachtung des Constraints durchgeführt werden; in Kapitel 4.5.3 wird gezeigt, in welcher Weise die Transformationsphase durch die Constraints Anforderung und Sequenz beeinflusst wird.

4.4.4 Verfahren bei der Konfiguration von SPF-Typ-Graphen

Auf Basis der Graphgrammatik beginnt jede Konfiguration eines SPF-Typ-Graphen mit dem Startsymbol, bei dem es sich um den Nichtterminalknoten der Wurzel des Graphen handelt. Je nachdem, wie viele Regeln zur Umwandlung eines Nichtterminalknotens existieren, müssen Fachanwender selbst Konfigurationsentscheidungen treffen oder die Konfiguration von Teilgraphen kann automatisch ohne Interaktion mit dem Modellierer erfolgen. In diesem Kapitel wird beschrieben, wann eine automatisierte Konfiguration stattfinden kann, wie diese funktioniert und welche Möglichkeiten der manuellen Konfiguration existieren.

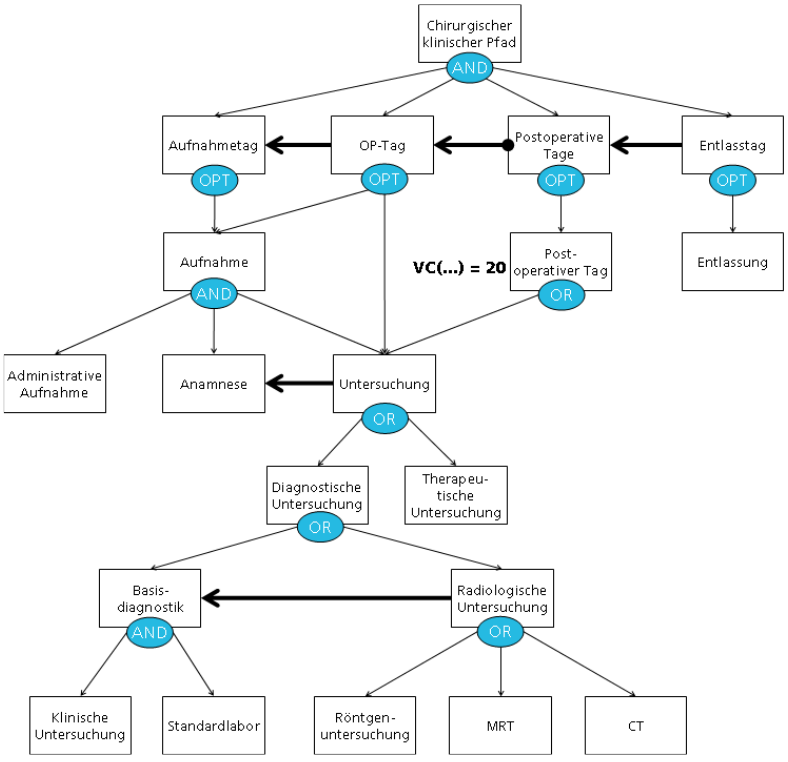
4.4.4.1 Automatisierte Konfiguration

Nach der Generierung einer Menge von Produktionsregeln auf Basis eines SPF-Typ-Graphen, ist es nun möglich, durch gezielte Anwendung der Regeln SPF-Graphen zu bilden, die anschließend in konkrete Prozessdefinitionen umgewandelt werden können. Dabei müssen nicht alle Konfigurationsentscheidungen manuell von den Fachanwendern getroffen werden. Stattdessen kann die Auswahl von Produktionsregeln automatisiert werden; dies geschieht immer dann, wenn bestimmte Knoten verpflichtend in SPF-Graphen vorkommen müssen und keine

Konfigurationsalternativen bereitstehen, die die Entscheidung eines Fachanwenders erforderlich machen.

In welchem Maße die automatisierte Konfiguration den Aufwand für den gesamten Konfigurationsprozess reduzieren kann, soll an einem Beispiel demonstriert werden. Die nachfolgende Abbildung zeigt den Ausschnitt aus einem SPF-Typ-Graphen für chirurgische klinische Pfade. Angenommen, eine Konfigurationseinstellung entspricht einer Regelanwendung; müsste jeder Knoten, unabhängig von dem ihm zugeordneten Operator und seiner Kardinalität manuell konfiguriert werden, können in dem Beispiel bereits bis zu 545 Einstellungen erforderlich sein, wenn tatsächlich das Maximum aller Knoten selektiert wird. Allein für den Knoten »Post-operativer Tag« kommt es zu insgesamt 39 Regelanwendungen, wenn tatsächlich 20 Klone des Knoten gebildet werden; 440 Regelanwendungen ergeben sich, wenn am Aufnahmetag, Operationstag und an jedem post-operativen Tag je eine komplette Untersuchung (bestehend aus allen selektierbaren Knoten) durchgeführt wird. An diesem Beispiel wird rasch deutlich, dass einem Mechanismus zur Automatisierung der Konfiguration eine hohe Bedeutung zukommt. Im Folgenden soll daher untersucht werden, unter welchen Voraussetzungen sich der Konfigurationsprozess automatisieren lässt; schließlich wird ein entsprechender Algorithmus spezifiziert.

Abbildung 74 Ausschnitt aus einem SPF-Typ-Graphen für chirurgische klinische Pfade



Für alle Knoten mit Knotenkardinalität *eins* werden bereits mehrere Produktionsregeln in Abhängigkeit der Nachfolger erzeugt; wobei nur maximal eine davon manuell angewendet werden kann. Z. B. existieren für den Knoten »Untersuchung« die folgenden Regeln:

$N_{Untersuchung} \rightarrow N^{1}_{Untersuchung}$

$N^{1}_{Untersuchung} \rightarrow Untersuchung\ (N_{Diagnostische\ Untersuchung})$

$N^1_{Untersuchung} \rightarrow Untersuchung\ (N_{Therapeutische\ Untersuchung})$
$N^1_{Untersuchung} \rightarrow Untersuchung\ (N_{Diagnostische\ Untersuchung}, N_{Therapeutische\ Untersuchung})$

Da keine Auswahlmöglichkeit zwischen mehreren Klonen gegeben ist, kann das Nichtterminalsymbol $N_{Untersuchung}$ sofort nach Erscheinen im SPF-Graph automatisch in das Nichtterminalsymbol $N^1_{Untersuchung}$ umgewandelt werden. Auf diese Weise lässt sich die Anzahl der Einstellungsmöglichkeiten von zwei Regelanwendungen auf eine reduzieren. Bei Knoten ohne Nachfolger ergibt sich ebenfalls Automatisierungspotential. Betrachten wir den Knoten »MRT«; entschließt sich die Fachanwenderin während der Konfiguration des Knotens »Radiologische Untersuchung«, dass ein MRT durchgeführt werden soll, so sollte für die anschließende Konfiguration des MRT-Knotens selbst keine manuelle Einstellung mehr notwendig sein; es existieren jedoch zwei Regeln zur Konfiguration von »MRT«:

$N_{MRT} \rightarrow N^1_{MRT}$
$N^1_{MRT} \rightarrow MRT$

Erscheint im SPF-Graph das Nichtterminalsymbol N_{MRT}, so lässt sich dieses automatisch in das Terminalsymbol *MRT* transformieren. Auf diese Weise entfallen zwei manuelle Regelanwendungen. Auch die Ausführung von Produktionsregeln für Knoten, die die Kardinalität *eins* besitzen und denen als logische Verknüpfung der AND-Operator zugeordnet wurde, lässt sich automatisieren. In solchen Fällen können alle nachfolgenden Knoten selektiert werden, ohne dass dafür eine weitere Nutzerinteraktion notwendig wäre. Betrachten wir zu diesem Zweck die Regelmenge des Knotens »Basisdiagnostik«:

$N_{Basisdiagnostik} \rightarrow N^1_{Basisdiagnostik}$
$N^1_{Basisdiagnostik} \rightarrow Basisdiagnostik\ (N_{Klinische\ Untersuchung}, N_{Standardlabor})$

Analog zu Knoten mit Kardinalität *eins* und Knoten ohne Nachfolger kann das Nichtterminalsymbol $N_{Basisdiagnostik}$ durch die automatische Auswertung von zwei Produktionsregeln in den Term $Basisdiagnostik\ (N_{Klinische\ Untersuchung}, N_{Standardlabor})$ umgewandelt werden. In analoger Weise lässt sich auch die Konfiguration eines Knoten, dem ein OR- oder XOR-Operator zugeordnet ist und der über nur einen Nachfolger verfügt, vollständig automatisieren. Dieser Zusammenhang wird am Beispiel des Knoten »Postoperativer Tag« deutlich:

$N_{Postoperativer\ Tag} \rightarrow N^1_{Postoperativer\ Tag}$
$N^1_{Postoperativer\ Tag} \rightarrow Postoperativer\ Tag\ (N_{Untersuchung})$

Gemäß dem obigen Beispiel wären bei einer manuellen Konfiguration zwei Regelanwendungen durchzuführen, die bei einer Automatisierung komplett entfallen können. Nicht zuletzt lassen sich bei Knoten mit einer Knotenkardinalität größer *eins* viele Regelanwendungen automatisieren, sobald eine Fachanwenderin die Anzahl der Klone manuell bestimmt hat. Für den Knoten »Postoperative Tage« existieren die folgenden Produktionsregeln:

$N_{Postoperative\ Tage} \rightarrow N^1_{Postoperative\ Tage}$
$N_{Postoperative\ Tage} \rightarrow N^2_{Postoperative\ Tage}$
...
$N_{Postoperative\ Tage} \rightarrow N^{20}_{Postoperative\ Tage}$
$N^1_{Postoperative\ Tage} \rightarrow Postoperative\ Tage$
$N^1_{Postoperative\ Tage} \rightarrow Postoperative\ Tage\ (N_{Postoperativer\ Tag})$
$N^2_{Postoperative\ Tage} \rightarrow N^1_{Postoperative\ Tage}, N^1_{Postoperative\ Tage}$
...
$N^{20}_{Postoperative\ Tage} \rightarrow N^1_{Postoperative\ Tage}, N^{19}_{Postoperative\ Tage}$

Während sich Fachanwender weiterhin manuell für die Anzahl der Klone entscheiden müssen, kann die Auflösung von Nichtterminalsymbolen der Form $N^{x}_{Postoperative\ Tage}$ mit $x > 1$ jedoch automatisch erfolgen. Auf diese Weise lassen sich in dem Beispiel pro Knoten bis zu 18 Interaktionen mit den Benutzern einsparen.

Mit einem Algorithmus, der gezielt die genannten Automatisierungsoptionen bei der Konfiguration überprüft und ausnutzt, lässt sich die maximale Anzahl aller Einstellungsmöglichkeiten in dem Beispiel aus Abbildung 74 von 545 auf 77 reduzieren, wenn immer die höchste Anzahl an Klonen gewählt wird und jeder Klon einzeln konfiguriert wird. Wird hingegen die Anzahl der Klone auf ein oder zwei Knoten beschränkt oder Verfahren zur einheitlichen Konfiguration von Klonen eingesetzt (siehe Kapitel 4.4.4.2), reduziert sich der Konfigurationsaufwand auch bei einer hohen Anzahl an Klonen auf einige wenige Einstellungen. Der nächste Codeabschnitt entspricht der Spezifikation des Algorithmus in Pseudocode. Als Parameter wird dem Algorithmus die Graphgrammatik und der zu transformierende SPF-Graph übergeben.

Formel 15 Algorithmus für die automatisierte Konfiguration

```
startAutoConfiguration(GG, SG) = {
      restart ≔ false;
      FOR (i = 0; i < |V^SG_NT|; i + 1)
         v^SG ≔ V^SG_NT[i];
         rules = {(v^STG, L, R) ∈ GG.PR|V^L = v^SG};
         IF (|rules| = 1)
            applyRule(SG, GG, v^SG, rules[0]); //Regel nur auf identifizierten
Knoten anwenden
            restart ≔ true;
         END-IF
      END-FOR
      IF restart = true
         startAutoConfiguration(GG, SG); //rekursiver Aufruf
      END-IF
}
```

Der Funktionsalgorithmus ist sehr einfach, da allen geschilderten Anwendungsfällen die Bedingung gemein ist, dass nur eine Möglichkeit der Transformation (bzw. nur eine rechte Seite der Regel) existieren darf. Ist dies der Fall, kann die Umwandlung automatisiert durchgeführt werden. Für jeden nicht-terminalen Knoten im SPF-Graphen wird geprüft, ob eine Produktionsregel existiert, deren linke Seite dem nicht-terminalen Knoten im SPF-Graphen entspricht und für die es exakt eine rechte Seite gibt. Gilt diese Bedingung, kann über *applyRule* die Produktionsregel ausgeführt werden (siehe Kapitel 4.4.2.2). Die Funktion *startAutoConfiguration* wird das erste Mal nach der Initialisierung des SPF-Graphen aufgerufen; anschließend wird sie jedes Mal nach dem manuellen Auslösen einer Produktionsregel gestartet. Außerdem gibt es einen Bool'schen Wert *restart*, der auf »wahr« gesetzt wird, wenn sich die Struktur des SPF-Graphen aufgrund einer automatisierten Regelanwendung verändert; dies bedeutet, dass die Funktion abschließend in einem rekursiven Prozess erneut durchlaufen wird. Bei dem rekursiven Aufruf werden dem Algorithmus wieder die Graphgrammatik und der veränderte SPF-Graph übergeben. Der Algorithmus terminiert spätestens bei der vollständigen Konfiguration des SPF-Graphen, da der Graph dann über keine nicht-terminalen Knoten mehr verfügt.

4.4.4.2 Manuelle Konfiguration

Üblicherweise entspricht die Konfiguration von SPF-Typ-Graphen einem »Top-down«-Vorgehen. Das bedeutet, dass Produktionsregeln in ihrer vorgesehenen Reihenfolge beginnend bei dem Startsymbol der Graphgrammatik und fortschreitend über die Zweige der Teilgraphen bis hin zu den SPF Knoten selektiert und angewandt werden. Wie sich diese Top-down-Strategie gegenüber den Fachanwendern äußert, soll am Beispiel des SPF-Typ-Graphen aus Abbildung 74 kurz erläutert werden. Dabei werden bereits Möglichkeiten der automatischen Konfiguration mit berücksichtigt.

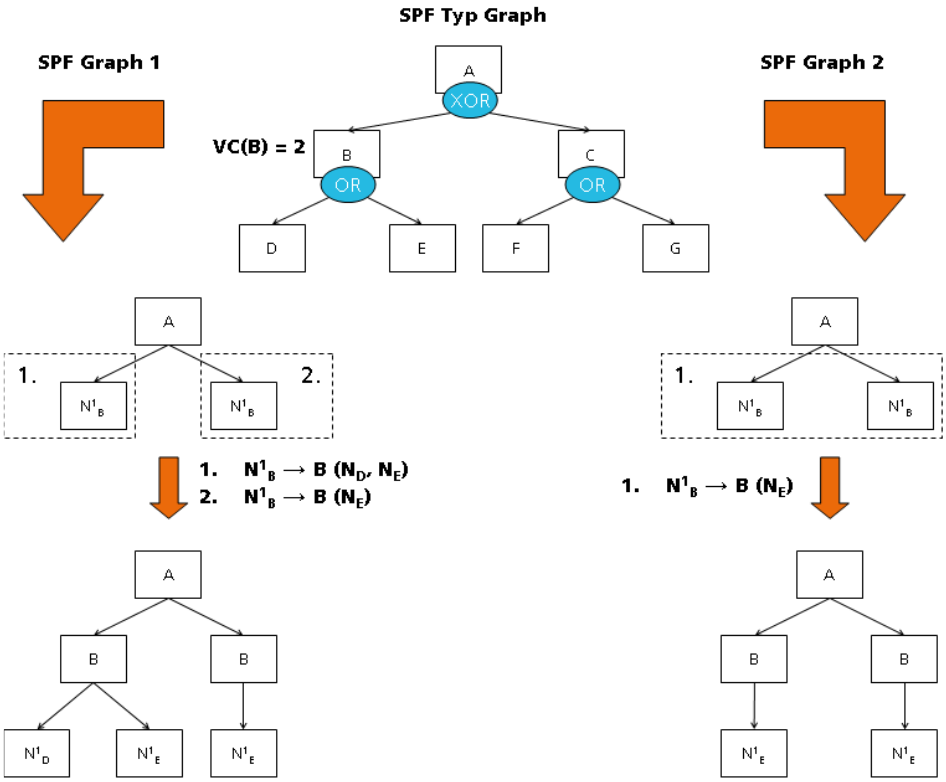
Jeder SPF-Graph zu einem chirurgischen klinischen Pfad setzt sich aus den Knoten »Aufnahmetag«, »Operationstag«, »Postoperative Tage« und »Entlasstag« zusammen. Da es auch die Möglichkeit der ambulanten Operation gibt, können alle Knoten unterhalb von »Aufnahmetag«, »Postoperative Tage« und »Entlasstag« optional gewählt werden. Hat sich eine Fachanwenderin also für das Anlegen eines chirurgischen klinischen Pfades entschieden, so besteht der SPF-Graph aus dem Wurzelknoten und vier nachfolgenden Knoten. Beschließt sie nun, dass es am Operationstag eine Aufnahme geben soll, so schließt dieser Knoten automatisch die »Administrative Aufnahme«, die »Anamnese« und die »Untersuchung« mit ein. Bei der Konfiguration des Knotens »Untersuchung« selektiert sie »Diagnostische Untersuchung« und »Therapeutische Untersuchung«. Die diagnostische Untersuchung wird so verfeinert, dass die Maßnahme sowohl die »Basisdiagnostik« als auch eine »Röntgenuntersuchung« als radiologische Untersuchung umfasst. Der Pfad soll nur einen postoperativen Tag enthalten, an dem noch einmal eine Untersuchung bestehend aus »Basisdiagnostik« und »Röntgenuntersuchung« zur Kontrolle vorgesehen sind.

Durch die Gleichartigkeit, in der die diagnostische Untersuchung am Operationstag und am postoperativen Tag gestaltet wird, ergibt sich bei der manuellen Konfiguration zusätzliches Optimierungspotential. Angenommen, die Fachanwenderin konfiguriert den SPF-Typ-Graphen derart, dass im SPF-Graphen zwei nicht-terminale Knoten vom Typ »Diagnostische Untersuchung« vorkommen. Unter Nutzung der in Kapitel 4.4.2.2 spezifizierten Funktion *applyRule* ist es nun möglich, dass beide Klone gleichzeitig konfiguriert werden. In diesem Fall muss die Fachanwenderin lediglich die Möglichkeit haben, entweder die beiden nicht-terminalen Knoten zusammen auszuwählen oder sie kann einstellen, dass alle aktuellen Klone dieser Art gleichförmig konfiguriert werden sollen. Das heißt, *applyRule* wird in Abhängigkeit der selektierten Knotenmenge mehrmals in Folge aufgerufen. Bei der Festlegung der Knotenmenge, auf die dieselbe Produktionsregel angewendet werden soll, lassen sich also prinzipiell folgende Optionen unterscheiden:

- **Bedingung mit Bezug zu der eindeutigen Identität eines Knoten im SPF-Graphen:** Diese Bedingung drückt aus, dass die Regel auf einen Knoten angewandt werden soll, der an Hand seiner Knotenbezeichnung eindeutig identifiziert werden kann; in diesem Fall hat der Fachanwender also einen bestimmten Knoten im SPF-Graphen ausgewählt, den er konfigurieren möchte. Dies entspricht auch der Standardsituation.
- **Bedingung mit Bezug zu der linken Seite einer Produktionsregel:** Statt jeden Knoten einzeln zu konfigurieren, ist es auch möglich, dieselbe Produktionsregel gleichzeitig auf mehrere nicht-terminale Knoten anzuwenden. Bedingung hierfür ist, dass alle diese Knoten der linken Seite der gewünschten Produktionsregel entsprechen. D. h. zwei nicht-terminale Knoten mit der derselben Bezeichnung N^1_A können gemeinsam konfiguriert werden; dies ist jedoch nicht möglich, wenn der eine Knoten die Bezeichnung N^1_A und der andere die Bezeichnung N_A trägt.

Die folgende Abbildung demonstriert die beiden Möglichkeiten der Konfiguration an einem Beispiel.

Abbildung 75 Optionen bei der Konfiguration von Klonknoten



Bei SPF-Graph 1 sollen die Klone von *B* unterschiedlich konfiguriert werden. Aus diesem Grund muss ihre Identifikation des jeweiligen Klonknotens an Hand seiner eindeutigen Identität erfolgen. Anschließend wird die Produktionsregel nur auf diesen einzelnen Knoten angewandt. Im zweiten Fall sollen alle Klone gleichartig konfiguriert werden. Hierbei ist eine Aufwandsersparnis möglich, indem die Produktionsregel nur einmal ausgewählt werden muss, um sie auf beide Klone anzuwenden. Diese Möglichkeit besteht nicht nur bei Klonen mit gemeinsamem Vorgänger, sondern auch bei Klonen in Folge von Mehrfachreferenzen. Die gleichartige Konfiguration von Klonen birgt großes Optimierungspotential speziell im Gesundheitswesen. Dort gibt es viele Prozessteile, die regelmäßig wiederholt werden oder zumindest gleichförmig ablaufen. Dazu zählen z. B. Aktivitäten wie Visite, Pflege oder Mobilisation an postoperativen Tagen.

Bei der Konfiguration von Knoten durch den Aufruf der Funktion *applyRule*, können also zwei Fälle unterschieden werden. Entweder werden die zu konfigurierenden Knoten an Hand ihrer Knotenidentität eindeutig festgelegt oder es handelt sich um alle Knoten, die dem Abbild der linken Seite der Produktionsregel entsprechen. Die Menge aller Klonknoten lässt sich sehr einfach an Hand des Graphmorphismus ermitteln, der zu der linken Seite einer Produktionsregel alle Abbilder im SPF-Graphen identifiziert. Sei also $GG = (START, PR, STG)$ eine Graphgrammatik, $rule \in PR^{GG}$ die für einen Knoten ausgewählte Regel und $SG = (V^{SG}, E^{SG}, V^{SG}_{T}, V^{SG}_{NT}, ...)$ der zu konfigurierende SPF-Graph. Das folgende Codesegment zeigt die Einbettung

der Funktion *applyRule* in einer Schleife über alle Teilgraphen im SPF-Graphen, die der linken Seite der Produktionsregel entsprechen.

Formel 16 Gleichförmige Konfiguration von Klonknoten

```
//Ermittlung aller Abbilder der linken Seite im gegebenen SPF-Graphen SG
L* = o(L^rule);
//Schleife über L*
FOR (i = 0; i < |L*|; i + 1)
   L = L*[i];
   applyRule(GG, SG, V^L, rule);
END FOR
```

Ebenso ist es natürlich möglich, eine Schleife über eine vorab selektierte Menge von nicht-terminalen Knoten laufen zu lassen, wenn nur eine Auswahl von Klonknoten gleichartig konfiguriert werden soll. Bedingung für die gleichartige Konfiguration von Knoten ist allerdings, dass es sich tatsächlich immer um dasselbe nicht-terminale Symbol handelt. Sei $v_1, v_2, ..., v_n \in V^{SG}_{NT}$ mit $n \in \mathbb{N}$ die Menge der gleichartig zu konfigurierenden Knoten; dann muss gelten:

$$\forall v_1, v_2, ..., v_n \in V^{SG}_{NT} \; \exists (v, L, R) \in PR^{GG}: v_1 = v_2 = ... = v_n = V^L$$

Verschiedene Möglichkeiten, wie eine Auswahl gleichartiger nicht-terminaler Knoten im SPF-Graphen automatisch ermittelt werden kann, werden in Kapitel 4.6.3 vorgestellt. Wie bei der manuellen Konfiguration einzelner Knoten gilt jedoch auch für diese Art der Konfiguration, dass die grundlegende Top-down-Strategie einzuhalten ist. Trotz der Option zur gleichzeitigen Konfiguration von Klonen kann das Top-down-Vorgehen in bestimmten Situationen zu umständlich sein und Fachanwender unnötig Zeit kosten. Angenommen, ein Fachanwender entscheidet sich zu Beginn der Konfiguration noch nicht, ob ein separater Aufnahmetag stattfinden soll, oder ob die notwendigen Untersuchungen stattdessen direkt am OP-Tag durchgeführt werden. Es soll jedoch festgelegt werden, dass die diagnostische Untersuchung immer mindestens die Basisdiagnostik umfasst, egal an welchem Tag sie durchgeführt wird. Eine Top-down-Konfiguration kommt in diesem Fall also nicht in Frage. Stattdessen muss der Möglichkeitsraum, der durch die Produktionsregelmenge der Graphgrammatik aufgespannt wird, durch einen »Bottom-up«-Ansatz verändert werden. Modifikationen an der Graphgrammatik sind allerdings nicht erlaubt, da dann die Korrektheitskriterien, die für SPF-Typ-Graphen gelten, nicht überprüft werden können und es zu gravierenden Problemen bei der Konfiguration kommen kann. Stattdessen ist es erforderlich, den SPF-Typ-Graphen selbst zu modifizieren. Da SPF-Typ-Graphen jedoch sehr umfangreich werden können, sollte vorher festgelegt werden, welche Teile des SPF-Typ-Graphen durch Fachanwender vor der Konfiguration verändert werden dürfen. Darüber hinaus können die zulässigen Änderungsoperationen so definiert werden, dass nur Einschränkungen im Vergleich mit dem originären SPF-Typ-Graphen gestattet sind, aber keine Erweiterungen (z. B. durch das Einfügen neuer Kanten und Knoten oder das Heraufsetzen von Kardinalitäten). Der Bottom-up-Ansatz basiert dann auf Operationen, die auch zur Erstellung von SPF-Typ-Graphen verwendet werden, deren Spezifikation jedoch nicht Teil dieser Arbeit ist.

4.4.5 Zusammenfassung der Ergebnisse

Nach der Identifikation der wesentlichen ProzessfragmSPOT-Domain-Model-Configuratorente und der Verstetigung des domänenorientierten Prozesswissens durch Zuordnung der Frag-

mente zu Knoten eines SPF-Typ-Graphen, lassen sich klinische Pfade als Prozesse für bestimmte Behandlungsfälle erstellen. Dies geschieht durch Konfiguration des SPF-Typ-Graphen; bei den dabei entstehenden SPF-Graphen handelt es sich um typisierte Graphen, deren Elemente über einen Graphmorphismus auf die Elemente des SPF-Typ-Graphen abgebildet werden können. Um die Konformität der SPF-Graphen mit den Vorgaben aus dem SPF-Typ-Graphen im Sinne des »Correctness by Construction« Ansatzes sicherzustellen, wird der Konfigurationsvorgang formalisiert. Zu diesem Zweck wird ausgehend von einem SPF-Typ-Graphen eine Graphgrammatik automatisch generiert. Der SPF-Graph lässt sich anschließend durch Anwendung der Regeln der Graphgrammatik schrittweise entwickeln.

Mit Hilfe der Graphgrammatik ist es möglich, die Einhaltung der Vorgaben im Hinblick auf Hierarchie, logische Operatoren und Kardinalitäten sicherzustellen. Die Erfüllung der Constraint-Relationen lässt sich hingegen nur durch geeignete Strukturierung des SPF-Typ-Graphen auf diesem Wege erreichen. In dem Kapitel wurde gezeigt, wie die Deklaration von Constraints durch Anordnung der Knoten im SPF-Typ-Graphen vermieden werden kann. Dabei wurde auch festgestellt, dass diese Lösung sowohl aus fachlicher Sicht als auch im Hinblick auf Komplexität nicht immer adäquat ist. Aus diesem Grund wurde ein Verfahren entwickelt, mit dem die Einhaltung von Constraint-Relationen automatisch sichergestellt werden kann. Die entsprechenden Algorithmen sind ebenfalls Teil der Arbeitsergebnisse.

Schließlich wurden unterschiedliche Verfahren zur Konfiguration von SPF-Typ-Graphen vorgestellt. So wurde gezeigt, dass sich Methoden zur automatischen Konfiguration sehr einfach realisieren lassen und gleichzeitig zu einer hohen Reduktion des manuellen Konfigurationsaufwandes führen können. Darüber hinaus wurden Verfahren zur gleichartigen Konfiguration von Klonen entwickelt, die ebenfalls entscheidend dazu beitragen, den Aufwand zur Erstellung neuer Prozessdefinitionen zu verringern. Insgesamt wurden die folgenden Resultate erreicht:

- Formale Spezifikation von SPF-Graphen als Ergebnis der Konfiguration von SPF-Typ-Graphen
- Entwicklung eines Algorithmus zur automatischen Generierung von Graphgrammatiken auf Basis von SPF-Typ-Graphen
- Erstellung eines Formalismus zur Konfiguration von SPF-Typ-Graphen durch Anwendung der Regeln der Graphgrammatik
- Entwicklung von Algorithmen zur Sicherstellung der Einhaltung von Constraint-Relationen
- Konzeption von Optimierungsstrategien im Hinblick auf den Konfigurationsvorgang (automatische Konfiguration und gemeinsame Konfiguration von Klonen)

4.5 Transformationsphase

Die Transformationsphase markiert den Übergang vom sprachunabhängigen SPF-Typ-Graphen und dem SPF-Graphen als typisierter, konkreter Ausprägung eines klinischen Pfades, hin zu einer Prozessdefinition, die auf einer definierten Modellierungssprache beruht. Bei dieser Modellierungssprache kann es sich z. B. um SPOT-ML (SPOT Modeling Language) als die graphische Repräsentationsform für SPOT-MM, um Petri Netze, BPMN, UML Aktivitätsdiagramme, EPC, BPEL oder WSM Nets handeln. Aufgrund der Fülle an möglichen Zielsprachen basiert die Transformationsphase also immer auf der Durchführung einer Reihe proprietärer Operationen, aus denen schließlich eine Prozessdefinition hervorgeht. Die Korrektheit der Prozessdefinition ist daher auch an die korrekte Art, Reihenfolge und Durchführung dieser Operationen gekoppelt. Betrachtet man SPOT-MM, so existiert für dieses Metamodell ein XML-Schema, das auch grundlegende Vorgaben zur Korrektheit der Prozessdefinition auf-

stellt. Die Prozessdefinition basierend auf SPOT-MM entspricht daher immer einem XML-Dokument; SPOT-ML definiert für die einzelnen XML-Elemente eine graphische Repräsentation, die in einem Modellierungswerkzeug dargestellt werden kann. Die Transformation des SPF-Graphen in eine Prozessdefinition, die zu SPOT-MM konform ist, könnte daher durch Aufrufe entsprechender XSLT Skripte oder durch Routinen einer beliebigen Programmiersprache erfolgen. Bisher existiert für SPOT-MM jedoch keine API, die das automatische Generieren der XML-Dokumente ermöglicht; d. h. sämtliche XSLT Skripte oder Programmfunktionen müssten zunächst erzeugt werden, um in Kapitel 6 eine Validierung der hier vorzustellenden Konzepte zu erreichen, was mit erheblichem Aufwand verbunden ist.

Bei der Entwicklung des SPOT Healthcare Demonstrators wurde ADEPT2 bzw. die kommerzielle Version Aristaflow® BPM Suite als Basis zur Ausführung von Prozessdefinitionen genutzt (siehe Kapitel 4.1.1). Der SPOT Compiler als Bestandteil des Demonstrators und Kern der SPOT Architektur ist bereits in der Lage, Prozessdefinitionen basierend auf SPOT-MM in eine WSM Net Repräsentation zu überführen, die von den Komponenten der BPM Suite interpretiert und verarbeitet werden kann. Im Gegensatz zu den APIs vieler anderer Prozessmodellierungswerkzeuge bietet die ADEPT2 API komplexe Änderungsoperationen an, die semantisch höherwertige Modifikationen an Prozessdefinitionen realisieren (siehe Kapitel 3.2.6). Z. B. ist das Einfügen einer einzelnen Prozessaktivität in ADEPT2 nicht möglich; stattdessen kann eine Aktivität sequentiell, parallel oder bedingt zu einer Menge bereits existierender Aktivitäten in die Prozessdefinition eingefügt werden. Da ADEPT2 die Durchführung von Änderungsoperationen nur unter der Voraussetzung gestattet, dass die so entstehende Prozessdefinition die definierten Korrektheitskriterien erfüllt, wird vor der Durchführung einer Änderungsoperation geprüft, ob diese im Hinblick auf den Kontrollfluss überhaupt zulässig ist. Verstöße gegen die Korrektheit des Datenflusses sind erlaubt; sie werden jedoch automatisch erkannt, so dass verhindert werden kann, dass eine Prozessdefinition auf dem Prozess-Server deployed und instanziiert wird, bei der die Datenversorgung der Aktivitäten nicht sichergestellt ist.

Aufgrund des hohen Entwicklungsgrades von ADEPT2, der komfortablen Nutzbarkeit der API und den positiven Erfahrungen während der Entwicklung des SPOT Healthcare Demonstrators, orientiert sich die Beschreibung der Transformationsphase in dieser Arbeit an den konzeptionellen Grundlagen von ADEPT2 und den dort verwendeten WSM Nets. Auch die Evaluierung wird am Beispiel von ADEPT2 durchgeführt, indem der SPOT-Process-Compiler so angepasst wird, dass er SPF-Graphen in WSM Net Prozessdefinitionen überführen kann. Ein wichtiger Vorteil in der Nutzung dieses WfMS ist auch in der Formalisierung der Prozessmodelle als ADEPT2 WSM Nets begründet; lassen sich SPF-Graphen in gültige WSM Nets transformieren, so liefert dies gleichzeitig einen Nachweis für die strukturelle Integrität der Graphen. Da die Prozessentwicklung mit WSM Nets dem prozeduralen Prozessmodellierungsparadigma entspricht, leistet die Spezifikation der Transformationsphase am Beispiel von ADEPT2 auch einen Beitrag zu der Problemstellung, in welcher Weise sich deklarative Modelle, wie SPF-Graphen, als prozedurale Prozessgraphen darstellen lassen. Wie an Hand der Regeln für die Transformation in Kapitel 4.5.3 deutlich wird, kommt dem Konzept der Synchronisationskanten in ADEPT2, mit deren Hilfe das Gebot der strikten Blockstrukturierung gelockert werden kann, dabei eine besondere Bedeutung zu.

Zunächst erfolgt eine Einführung in die formalen Konzepte von ADEPT2, soweit das im Rahmen der Transformationsphase notwendig ist. Dazu zählen zum einen die Vorstellung von WSM Nets als Formalismus zur Erstellung von Prozessdefinitionen und zum anderen die Beschreibung der für die Transformation von SPF-Graphen in WSM Nets notwendigen ADEPT2-

Änderungsoperationen. Anschließend wird formal spezifiziert, wie SPF-Graphen durch Nutzung von Änderungsoperationen in die WSM Net Notation umgewandelt werden können. Bei der Transformation werden die SPF Knoten des SPF-Graphen direkt in Aktivitäten überführt, die durch die Prozessfragmente realisiert werden, die den SPF Knoten im SPF-Typ-Graphen zugeordnet sind. Die Struktur des Kontrollflusses, der die Ausführungsreihenfolge zwischen den Aktivitäten definiert, wird durch die im SPF-Typ-Graphen spezifizierten Constraint-Relationen determiniert. Anschließend wird diskutiert, wie der Datenfluss aus der Schnittstellenspezifikation der Prozessfragmente abgeleitet werden kann. Zum Abschluss wird gezeigt, wie sich durch die Transformation unvollständig konfigurierter SPF-Typ-Graphen die von Weber et al. genannte zweite Klasse von »Change Pattern«, nämlich Änderungen in vordefinierten Regionen, umsetzen lassen [Weber et al. 2008].

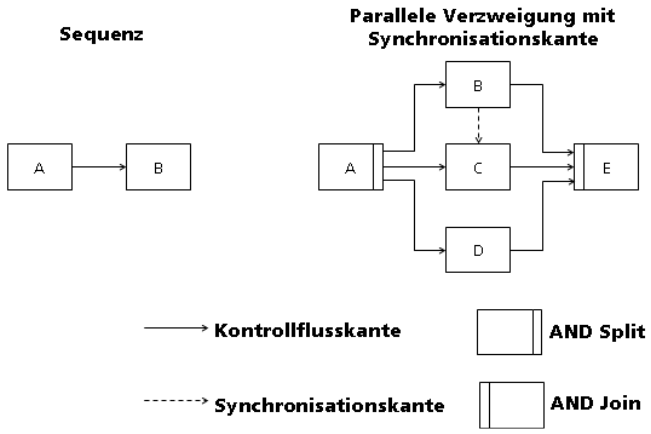
In diesem Kapitel wird allerdings kein Transformationsalgorithmus angegeben, da dieser lediglich Beispielcharakter mit Hinblick auf die Umwandlung nach WSM Nets hätte und nicht zu dem allgemeinen Lösungskonzept gehört. Bei der praktischen Evaluation des Lösungskonzeptes in Kapitel 1 wurde der bereits existierende SPOT-Process-Compiler allerdings um einen solchen Algorithmus erweitert; dieser basiert auf den in Kapitel 4.5.3 vorgestellten formalen Umwandlungsregeln.

4.5.1 Prozessdefinitionen als ADEPT2 WSM Nets und deren Korrektheitskriterien

Bei ADEPT2 handelt es sich um ein Workflow Management System, das auf einem formalen Modell für Prozessdefinitionen (ADEPT Basismodell bzw. WSM Nets) und einer Menge von komplexen Änderungsoperationen basiert. Änderungsoperationen entsprechen in ADEPT2 der konkreten Realisierung einer Vielzahl von Change Pattern, wie sie in [Weber et al. 2007, Weber et al. 2008] spezifiziert sind. Die Anwendung der Änderungsoperationen kann sowohl zum Zeitpunkt der Prozessmodellierung als auch während der Laufzeit des Prozesses erfolgen [Reichert & Dadam 1998]. In diesem Abschnitt werden die formalen Grundlagen von ADEPT2 erläutert, die benötigt werden, um den Vorgang der Transformation von SPF-Graphen in WSM Nets zu spezifizieren.

Bei WSM Nets handelt es sich um eine Sprache zur Beschreibung von Prozessdefinitionen, die im Rahmen des ADEPT bzw. ADEPT2 Projektes entwickelt wurde [Reichert & Dadam 1998]. Die formalen Grundlagen für WSM Nets wurden von Reichert in [Reichert 2000b] aufgestellt. WSM Nets repräsentieren gerichtete, strukturierte Graphen bestehend aus einer Aktivitäten- und einer Kantenmenge. Die Knoten eines Graphen können über sequentielle, parallele Kanten oder Synchronisationskanten miteinander verbunden sein (siehe Abbildung 76).

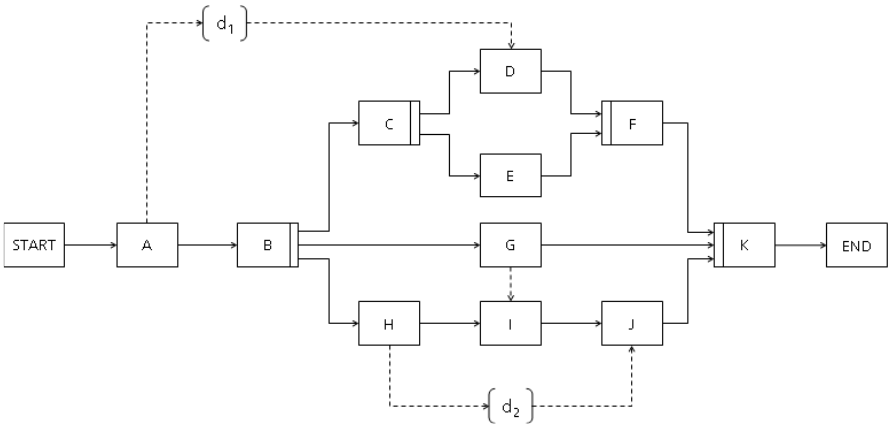
Abbildung 76 Transformationsrelevante Kontrollflusskonstrukte von WSM Nets



Neben den oben vorgestellten Konstrukten verfügen WSM Nets zusätzlich über spezielle Schleifenkonstrukte und bedingte Verzweigungen. Da sich statische Schleifen jedoch auch durch das sequentielle Hintereinanderschalten von Klonen realisieren lassen, wie in Kapitel 4.5.3 gezeigt wird, werden Schleifen im Rahmen der Transformationsphase nicht weiter berücksichtigt. Während der Rekonfigurationsphase lassen sich sogar dynamische Schleifen durch das Hinzufügen von Klonen zur Laufzeit des Prozesses umsetzen. Bedingte Verzweigungen in Prozessdefinitionen ermöglichen, dass nicht alle Entscheidungen über das weitere Vorgehen bei der Behandlung zur Modellierungszeit getroffen werden müssen, sondern auch zur Laufzeit evaluiert werden können; dies macht z. B. Sinn, wenn der Fortgang der Behandlung von den Ergebnissen vorangehender Prozessschritte abhängt. Im SPF-Graphen wird die noch bestehende Unsicherheit z. B. hinsichtlich der Wahl radiologischer Untersuchungen durch nicht-terminale Knoten repräsentiert, die erst noch konfiguriert werden müssen. Da das Wissen über die verschiedenen Alternativen also nicht im zu transformierenden SPF-Graphen verfügbar ist, sondern in dem jeweiligen SPF-Typ-Graphen und der Graphgrammatik, werden bei der Transformation von SPF-Graphen keine bedingten Verzweigungen generiert. Mögliche Lösungsansätze, wie Entscheidungen dennoch erst während der Ausführung der Prozessinstanz getroffen werden können, werden in Kapitel 4.5.5 dargestellt. Sie sind jedoch nicht Bestandteil des Kern-Transformationsprozesses, der sich ausschließlich auf terminale Knoten bezieht und der in dieser Arbeit spezifiziert werden soll.

Das bezeichnende Strukturmerkmal von WSM Nets ist die Anordnung der Aktivitäten in Kontrollblöcken; diese Vorgehensweise wurde bereits in Kapitel 3.2.6 skizziert. Die nachfolgende Abbildung zeigt ein Beispiel für ein einfaches und korrekt strukturiertes WSM Net. Der Datenfluss wird in WSM Nets durch gestrichelte Kanten symbolisiert, die zu Datenobjekten hin- oder wegführen.

Abbildung 77 Beispiel für ein einfaches WSM Net



Im Folgenden wird eine mengenbasierte Definition für WSM Nets angegeben, die auf die in dieser Arbeit benötigten Typen von Kontroll- und Datenflüssen reduziert wurde. Datenobjekte werden immer global für ein WSM Net definiert und können über Lese- und Schreibkanten mit den Prozessaktivitäten verknüpft werden.

Definition (WSM Net). Das Tupel *W = (N, D, NT, CtrlE, SyncE, DataE)* entspricht einem WSM Net, wenn die folgenden Eigenschaften erfüllt sind:
- *N* (endliche, nicht leere Menge von Aktivitäten)
- *D* (endliche Menge von Datenobjekten)
- *NT: N ↦ {StartFlow, EndFlow, Activity, AndSplit, AndJoin}* (Knotentypfunktion)
- *CtrlE ⊆ N x N* (endliche, nicht leere Menge von gerichteten Kontrollflusskanten zwischen Aktivitäten)
- *SyncE ⊆ N x N* (endliche Menge von gerichteten Synchronisationskanten zwischen Aktivitäten paralleler Zweige)
- *DataE ⊆ N x D x {read, write}* (endliche, nicht leere Menge von Datenflusskanten zwischen Aktivitäten und Datenobjekten)
- *Template: N ↦ Templates ∪ {None}* (Zuordnungsfunktion für Aktivitätsvorlagen)

Die einzelnen Elemente haben die folgende Bedeutung:
- *N* entspricht einer endlichen Menge von Aktivitätenbezeichnungen; da jedes WSM Net mindestens aus einer Start- und einer Endaktivität besteht, die über einen Kontrollfluss miteinander verbunden sind, ist die Menge nicht leer.
- *D* beschreibt eine endliche Menge von Bezeichnungen für Datenobjekte, die zusammen den Datenkontext eines Prozesses bilden. Jedes Datenobjekt hat einen Wertebereich und nimmt bei der Initialisierung einen Default-Wert an, der im Standardfall dem Nullwert entspricht.
- *NT: N ↦ {StartFlow, EndFlow, Activity, AndSplit, AndJoin}* ist eine Funktion, die jeder Aktivität einen Typ zuordnet; *StartFlow* ist z. B. der Typ der Startaktivität, *EndFlow* der Typ der Endaktivität.
- *CtrlE ⊆ N x N* ist eine endliche, nicht leere Menge von gerichteten Kanten, über die zwei Aktivitäten innerhalb einer Prozessdefinition miteinander verknüpft sind, wobei auf die Einhaltung der Blockstruktur geachtet werden muss. Analog wie bei SPF-Typ-Graphen wird jeweils eine Quell- und eine Zielfunktion definiert:
  source: CtrlE ↦ N (Quellfunktion)
  target: CtrlE ↦ N (Zielfunktion)

- *SyncE* $\subseteq$ *N x N* ist eine endliche Menge von gerichteten Synchronisationskanten, die zwischen den Aktivitäten paralleler Zweige eine Ausführungsreihenfolge vorgeben. Die oben definierten Funktionen *source* und *target* gelten in analoger Weise auch für Synchronisationskanten.
- *DataE* $\subseteq$ *N x D x {read, write}* repräsentiert die endliche Menge der Datenflusskanten zwischen Aktivität und Datenobjekt. Über das Prädikat *{read, write}* wird festgelegt, ob es sich um einen lesenden oder schreibenden Zugriff auf das Datenobjekt handelt.
- *Template: N* $\nrightarrow$ *Templates* $\cup$ *{None}* entspricht einer Funktion, die jeder Aktivität eine Aktivitätsvorlage oder *None* zuordnet. Aktivitäten, die mit keiner Aktivitätsvorlage verknüpft sind, werden auch als »Nullaktivitäten« bezeichnet; dabei handelt es sich um Aktivitäten, die keine fachliche Bedeutung haben und ausschließlich der Festlegung der Ausführungsreihenfolge oder dem Erhalt der Blockstruktur dienen. Typischerweise sind Aktivitäten vom Typ *AndSplit* und *AndJoin* Nullaktivitäten. Die Bedeutung von Aktivitätsvorlagen wurde bereits in Kapitel 4.1.2 dargestellt. Um die Transformation von SPF-Graphen nach WSM Nets zu erreichen, muss demnach jede Aktivitätsvorlage durch die Implementierung eines Prozessfragments repräsentiert werden; es gilt:

  $\forall n \in N: Template(n) \neq None \Rightarrow \exists p \in ProcessfragmentIds: Template(n) = Implementation^{p}$

Die formale, mengenbasierte Definition von WSM Nets macht es immer noch möglich, Prozessdefinitionen zu erstellen, die gegen die strukturelle Integrität verstoßen, z. B. weil sie zu Verklemmungen führen können oder nicht alle Aktivitäten mit denen von ihnen benötigten Datenobjekten versorgt sind. Aus diesem Grund wurden für WSM Nets die im Folgenden aufgelisteten Korrektheitskriterien definiert [Reichert & Dadam 1998, Reichert 2000b]. Sei *W = (N, D, NT, CtrlE, SyncE, DataE)* ein WSM Net, dann gilt:

- *W* hat eine eindeutige Startaktivität *Start* mit *NT(Start) = StartFlow* und eine eindeutige Endaktivität *End* mit *NT(End) = Endflow*.
- Jede Aktivität mit Ausnahme von Start- und Endaktivität verfügt mindestens über eine eingehende und eine ausgehende Kante, so dass es keine isolierten Aktivitäten geben kann, die während der Ausführung des Prozesses nicht erreichbar sind.
- Ein Kontrollblock W_B *= (N, CtrlE, SyncE)* entspricht einem Teilgraphen von *W* bestehend aus einer Menge von über Kontrollflusskanten miteinander verbundenen Aktivitäten. Die Struktur jedes Kontrollblocks orientiert sich an einem Blockkonzept, nachdem Überlappungen unterschiedlicher Blöcke nicht erlaubt sind; es ist jedoch möglich, dass Blöcke ineinander geschachtelt werden.
- Bei jedem Teilgraphen von *W* handelt es sich um einen azyklischen Graphen; d. h. die Bildung von Schleifen mit Hilfe von Kanten der Menge *CtrlE* und *SyncE* ist unzulässig.
- Für jede Aktivität, die über eine lesende Datenflusskante mit einem Datenobjekt verbunden ist, muss gelten, dass das Datenobjekt über eine schreibende Datenflusskante mit einer Vorgängeraktivität verknüpft ist.
- Parallele Schreibzugriffe auf Datenobjekte sind generell nicht erlaubt. Dies wird verhindert, indem die Deklaration von Datenflusskanten zum Schreiben eines Datenobjektes ausgehend von den Aktivitäten unterschiedlicher paralleler Zweige nicht möglich ist. Eine Ausnahme besteht, wenn die Reihenfolge der Schreibzugriffe durch die Angabe von Synchronisationskanten eindeutig geklärt ist.

Bei der Umwandlung eines SPF-Graphen in ein WSM Net ist es wichtig, im WSM Net navigieren zu können. Zu diesem Zweck muss es möglich sein, festzustellen, welche Aktivitäten der Menge der Vorgänger bzw. Nachfolger einer gegebenen Aktivität angehören. Um also sowohl

die direkten als auch die indirekten Vorgänger und Nachfolger einer Aktivität ermitteln zu können, werden die folgenden Hilfsfunktionen spezifiziert:

Definition (Vorgängerfunktionen). Sei $W = (N, D, NT, ...)$ ein WSM Net und $n \in N$ eine Aktivität innerhalb der Prozessdefinition. Die Menge aller direkten Vorgänger von n kann über die Funktion *pred* festgestellt werden:
$pred: N \mapsto 2^N$
mit
$pred(n) = \{m \in N | \exists\ e \in CtrlE \cup SyncE: n \in target(e) \wedge m \in source(e)\}$

Die Menge aller direkten und indirekten Vorgänger von n ist folgendermaßen definiert:
$pred^*: N \mapsto 2^N$
mit
$pred^*(n) = \{m \in N | m \in pred(n) \vee (\exists u \in N: u \in pred(n) \wedge m \in pred^*(u))\}$

In analoger Weise kann die Definition der Nachfolgerfunktionen angegeben werden.

Definition (Nachfolgerfunktionen). Sei $W = (N, D, NT, ...)$ ein WSM Net und $n \in N$ eine Aktivität innerhalb der Prozessdefinition. Die Menge aller direkten Nachfolger von n kann über die Funktion *succ* festgestellt werden:
$succ: N \mapsto 2^N$
mit
$succ(n) = \{m \in N | \exists e \in CtrlE \cup SyncE: n \in source(e) \wedge m \in target(e)\}$

Die Menge aller direkten und indirekten Nachfolger von n ist folgendermaßen definiert:
$succ^*: N \mapsto 2^N$
mit
$succ^*(n) = \{m \in N | m \in succ(n) \vee (\exists u \in N: u \in succ(n) \wedge m \in succ^*(u))\}$

In vielen Zusammenhängen ist es von Bedeutung, ausgehend von einer beliebigen Aktivitätenmenge N^*, den minimalen Kontrollblock zu bestimmen, der alle Aktivitäten dieser Menge umfasst [Reichert 2000b]:

Definition (MinBlock-Funktion). Sei $W = (N, D, NT, ...)$ ein WSM Net und $\varnothing \neq N^* \subseteq N - \{n \in N | NT(n) = StartFlow \vee NT(n) = EndFlow\}$ eine beliebige, nicht leere Knotenmenge. Ein Kontrollblock $W_B = (N_B, D_B, NT_B, ...)$ heißt minimaler Kontrollblock bezüglich N^*, wenn folgende Bedingung erfüllt ist: $N^* \subseteq N_B \wedge (\nexists\ W_B^* = (N_B^*, D_B^*, NT_B^*, ...): N^* \subseteq N_B^* \subset N_B$
Die Abbildung $minBlock: 2^N \mapsto N \times N$, die zu jeder nicht leeren Teilmenge von $N - \{n \in N | NT(n) = StartFlow \vee NT(n) = EndFlow\}$ den Start- und Endknoten des minimalen, sie umschließenden Kontrollblocks zurückliefert, heißt »MinBlock-Funktion«.

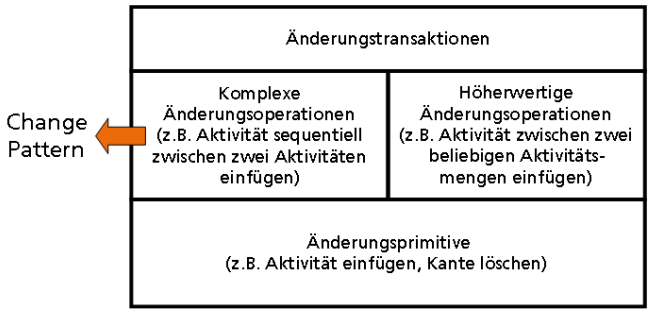
4.5.2 Vorstellung der Transformations-relevanten Änderungsoperationen

Change Pattern entsprechen vordefinierten Mustern, durch deren konkrete Realisierung sich neue Prozessdefinitionen entwickeln oder vorhandene Prozessdefinitionen und deren Instanzen anpassen lassen. Weber et al. unterscheiden zwei Arten von Change Pattern: Solche, mit denen sich ungeplante Ad-hoc-Änderungen durchführen lassen und solche, die eine flexible Variation vordefinierter Bereiche innerhalb einer Prozessdefinition ermöglichen [Weber et

al. 2007, Weber et al. 2008]. Neben der formalen Spezifikation von WSM Nets und deren Ausführungssemantik umfasst ADEPT2 auch eine Menge von komplexen Änderungsoperationen auf WSM Nets. Während diese Änderungsoperationen der Umsetzung der ersten Kategorie von Change Pattern dienen, entsprechen Konzepte wie Worklets (siehe 3.2.2) oder Pockets of Flexibility (siehe 3.2.3) möglichen Lösungen für die zweite Klasse von Pattern. In diesem Kapitel werden jedoch nur die Änderungsoperationen vorgestellt, die von ADEPT2 unterstützt werden und die für die Transformation von SPF-Graphen nach WSM Nets von Bedeutung sind. Darüber hinaus werden zusätzliche, hochwertige Änderungsoperationen von ADEPT2 eingeführt, die bisher nicht explizit in Form von Change Pattern generalisiert wurden, und um zwei eigene Änderungsoperationen ergänzt.

In ADEPT2 basieren sämtliche Änderungsoperationen auf so genannten »Änderungsprimitiven«; unter Änderungsprimitiven werden einfache Operationen z. B. zum Einfügen einer einzelnen Aktivität oder einer einzelnen Kontrollflusskante verstanden. Viele Modellierungswerkzeuge bieten nur diese Operationen zur Entwicklung und Anpassung von Prozessdefinitionen an. Da sich die Korrektheit einer Prozessdefinition jedoch nicht auf Grundlage von Änderungsprimitiven sicherstellen lässt, stellt die ADEPT2 API nur komplexe Änderungsoperationen bereit, die sich aus einer Menge von Änderungsprimitiven zusammensetzen; diese entsprechen der Umsetzung der Change Pattern für ungeplante Ad-hoc-Änderungen. Höherwertige Änderungsoperationen dienen z. B. dem Einfügen einer Aktivität zwischen zwei voneinander getrennt liegenden Mengen von Prozessaktivitäten. Änderungsoperationen lassen sich letztlich zu Änderungstransaktionen gruppieren; das bedeutet, dass eine Menge von Operationen logisch wie eine einzige Operation ausgeführt und auch in dieser Form rückgängig gemacht werden kann. Abbildung 78 illustriert den Zusammenhang zwischen Änderungsprimitiven, komplexen Änderungsoperationen, höherwertigen Änderungsoperationen und Änderungstransaktionen in ADEPT2.

Abbildung 78 Zusammenhang zwischen Änderungsprimitiven, komplexen Änderungsoperationen, höherwertigen Änderungsoperationen und Änderungstransaktionen in ADEPT2



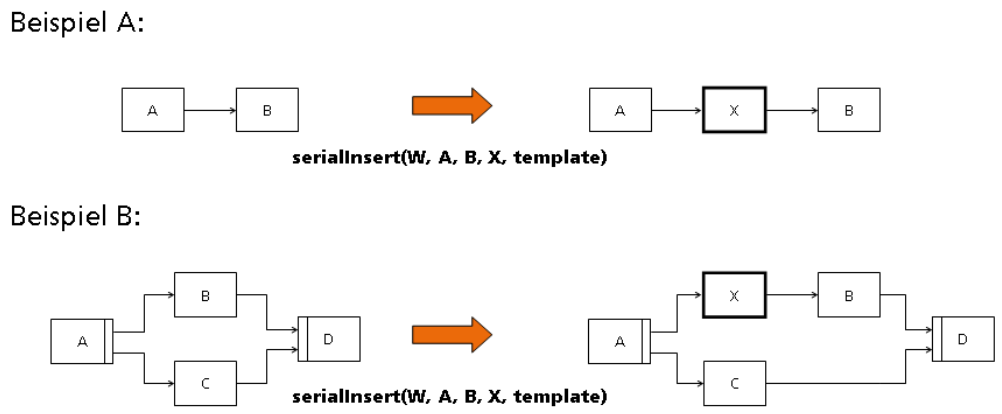
Der gesamte Transformationsprozess von SPF-Graphen in ein WSM Net lässt sich damit auf eine große Änderungstransaktion abbilden, die aus einer Vielzahl von Aufrufen von komplexen und höherwertigen Änderungsoperationen besteht. Während ADEPT2 eine breite Palette an komplexen Änderungsoperationen zur Verfügung stellt, die z. B. dem Hinzufügen, Entfernen, Verschieben oder Verschachteln von einzelnen Aktivitäten als auch vollständigen Kontrollblöcken dienen, werden im Rahmen der Transformation nur wenige Änderungsoperationen tatsächlich benötigt. Im Folgenden werden die während der Transformationsphase erforderlichen ADEPT2 Änderungsoperationen beschrieben. Eine detaillierte Spezifikation inklusive der entsprechenden Algorithmen findet sich in [Reichert 2000b].

Tabelle 18 Sequentielles Einfügen einer neuen Aktivität, Adaptation Pattern AP1 [Weber et al. 2008]

Änderungsoperation	Parameter		Semantik
	Name	Typ	
serialInsert(W, A, B, X, template)	W	WSM Net Prozessdefinition	Sequentielles Einfügen von Aktivität *X* zwischen zwei bereits im WSM Net *W* existierende Aktivitäten *A* und *B* Bedingungen: • $X \notin N$ • $A, B \in N$ • $B \in succ(A)$
	A	Vorgängeraktivität	
	B	Nachfolgeraktivität	
	X	Einzufügende Aktivität	
	template	Aktivitätsvorlage für *X*	

Im Fall des sequentiellen Einfügens wird eine neue Aktivität *X* zwischen zwei direkt aufeinanderfolgende Aktivitäten *A* und *B* eingefügt. Die nachfolgende Abbildung demonstriert die Auswirkungen dieser Änderungsoperation an zwei Beispielen.

Abbildung 79 Auswirkungen der ADEPT2-Änderungsoperation »serialInsert«



Bei dem sequentiellen Einfügen einer Prozessaktivität ist es prinzipiell irrelevant, ob die Aktivitäten *A* und *B* Bestandteil einer Sequenz oder einer parallelen Verzweigung sind. Zwingende Voraussetzung für die Durchführbarkeit der Operation ist lediglich, dass *A* und *B* in *W* direkt aufeinander folgen.

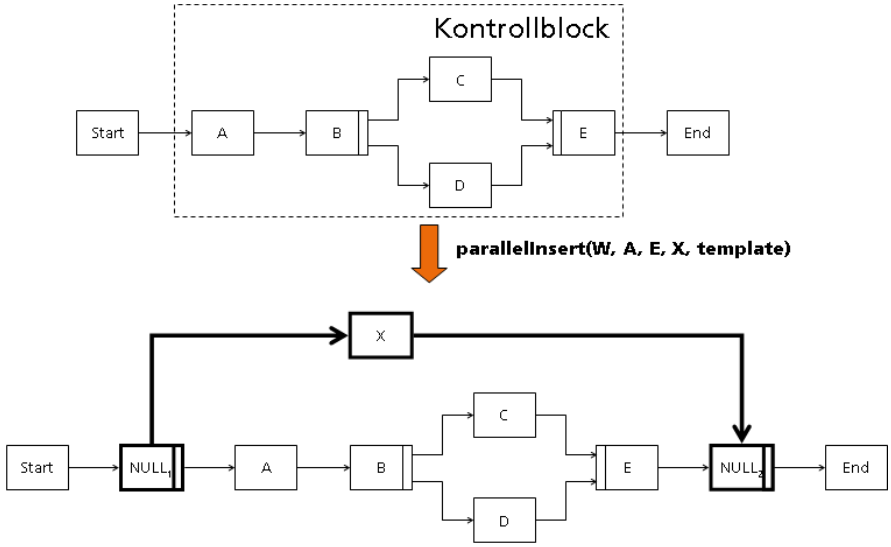
Tabelle 19 Paralleles Einfügen einer neuen Aktivität, Adaptation Pattern AP1 [Weber et al. 2008]

Änderungsoperation	Parameter		Semantik
	Name	Typ	
parallelInsert(W, A, B, X, template)	W	WSM Net Prozessdefinition	Einfügen von Aktivität *X* parallel zu einem Kontrollblock, dessen Beginn Aktivität *A* und dessen Ende Aktivität *B* entspricht Bedingungen: • $X \notin N$ • $A \in pred^*(B)$ • $A, B \in N - \{n \in N \mid NT(n) \in \{StartFlow,$
	A	Startaktivität markiert den Beginn des Kontrollblocks, zu dem parallel eingefügt werden soll	
	B	Endaktivität markiert das Ende des Kontrollblocks, zu dem parallel eingefügt werden soll	

	X	Einzufügende Aktivität	*EndFlow}} begrenzen einen Kontrollblock*
	template	Aktivitätsvorlage für *X*	

Beim parallelen Einfügen wird eine neue Aktivität *X* stets parallel zu einem existierenden Kontrollblock eingefügt, der durch die Aktivitäten *A* und *B* begrenzt ist. Intern fügt ADEPT2 vor Aktivität *A* einen Knoten vom Typ *AndSplit* ein, dem keine Aktivitätsvorlage zugeordnet ist; es handelt sich also um eine Nullaktivität. Analog dazu wird hinter Aktivität *B* ein Knoten vom Typ *AndJoin* angehängt, der ebenfalls eine Nullaktivität repräsentiert und lediglich dem Zusammenführen der parallelen Zweige dient, wobei der eine Zweig Aktivität *X* enthält und der andere den durch *A* und *B* begrenzten Kontrollblock umfasst. Die nachfolgende Abbildung demonstriert die Auswirkungen der Änderungsoperation an einem Beispiel.

Abbildung 80 Auswirkungen der ADEPT2-Änderungsoperation »parallelInsert«



Wichtig in Zusammenhang mit dem parallelen Einfügen ist die Feststellung, dass die Startaktivität *A* und die Endaktivität *B* so in der Prozessdefinition angeordnet sein müssen, dass die Schnittmenge aus *A* vereinigt mit seinen Nachfolgern und *B* vereinigt mit seinen Vorgängern einen korrekten Kontrollblock gemäß [Reichert 2000b] ergibt. Dementsprechend repräsentiert jeder Kontrollblock für sich genommen wieder ein korrektes WSM Net. Die Einhaltung der Blockstrukturierung bedeutet, dass es für jeden Verzweigungsknoten einen Synchronisationsknoten geben muss; umgekehrt muss für jeden Synchronisationsknoten ein Verzweigungsknoten existieren. Ist ein Verzweigungs-/Synchronisationsknoten Bestandteil eines Kontrollblockes, so umfasst der Block auch seinen Synchronisations-/Verzweigungsknoten.

Zu jedem Knoten im SPF-Graphen können Attribute angegeben werden, die zuvor im SPF-Typ-Graphen definiert wurden. Wurden den Attributen dort bereits Werte zugewiesen, so müssen die terminalen Knoten im SPF-Graphen diese Werte übernehmen, um den Graphmorphismus auf den Typgraphen zu erfüllen. Allen anderen Attributen müssen oder können Werte während der Konfiguration dynamisch zugewiesen werden (siehe Kapitel 4.4.1). Da die Möglichkeit besteht, den Aktivitäten innerhalb eines WSM Net beliebige Attribute zuzuordnen, erhalten sie die Attribute und Werte der terminalen Knoten im SPF-Graphen während der Transformation. Zu diesem Zweck kommt die Änderungsoperation »changeActivityAttribute« zum Einsatz.

Tabelle 20 Setzen bzw. anpassen der Attribute von Prozessaktivitäten

Änderungsoperation	Parameter		Semantik
	Name	Typ	
changeActivityAttribute(W, X, attribute, value)	W	WSM Net Prozessdefinition	Setzen des Attributes mit der Bezeichnung *attribute* des Fragments *X* auf den Wert *value* Bedingungen: • $X \in N$
	X	Aktivität deren Attribut geändert werden soll	
	attribute	Attributbezeichnung	
	value	Neuer Attributwert	

Bei der Änderungsoperation zum Setzen und Anpassen von Attributen handelt es sich um eine Operation, die nicht als Change Pattern in [Weber et al. 2007, Weber et al. 2008] identifiziert wurde. Beim Einfügen von Synchronisationskanten handelt es sich ebenfalls um eine ADEPT2 Änderungsoperation, die nicht als Change Pattern deklariert ist. Sie spielt jedoch bei der Transformation im Hinblick auf die Umsetzung höherwertiger Änderungsoperationen eine wichtige Rolle.

Tabelle 21 Einfügen von Synchronisationskanten zwischen Prozessaktivitäten paralleler Zweige

Änderungsoperation	Parameter		Semantik
	Name	Typ	
insertSyncEdge(W, A, B)	W	WSM Net Prozessdefinition	Einfügen einer Synchronisationskante von *A* nach *B* Bedingungen: • $A, B \in N$ • $(A, B) \notin SyncE$ • $B \notin pred^*(A)$ • $(Start, End) = minBlock(A, B):$ $NT(Start) = AndSplit \wedge NT(End) = AndJoin$
	A	Aktivität von der die Synchronisationskante ausgehen soll	
	B	Aktivität zu der die Synchronisationskante hinführen soll	

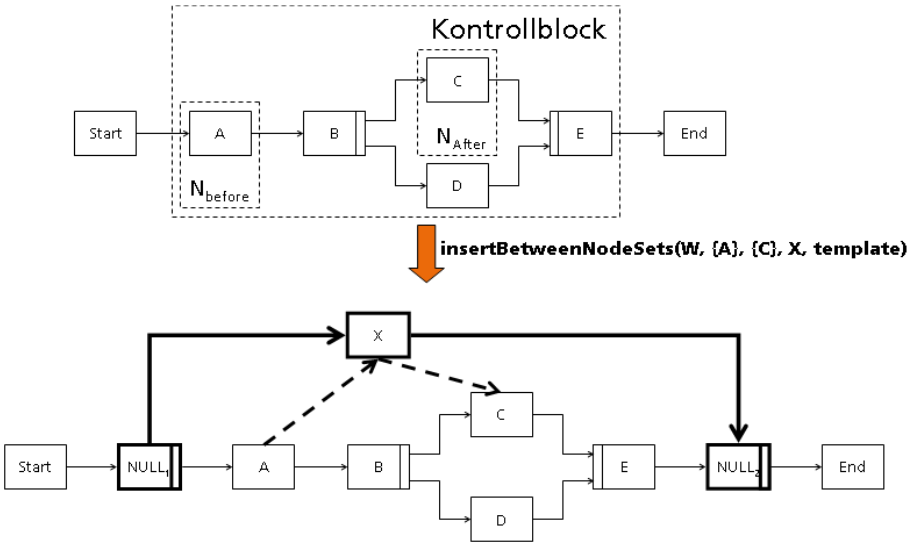
In ADEPT2 wird über die komplexen Änderungsoperationen hinaus die höherwertige Änderungsoperation »insertBetweenNodeSets« definiert, die es möglich macht, eine Aktivität zwischen zwei Mengen von Aktivitäten einzufügen, die die Kontrollblockstruktur nicht gezwungenermaßen berücksichtigen.

Tabelle 22 Einfügen einer neuen Aktivität zwischen zwei Aktivitätenmengen

Änderungsoperation	Parameter		Semantik
	Name	Typ	
insertBetweenNodeSets(W, N_{before}, N_{after}, X, template)	W	WSM Net Prozessdefinition	Paralleles Einfügen von Aktivität *X* und Festlegung der Reihenfolge durch Hinzufügen von Synchronisationskanten zwischen N_{before} und *X* bzw. *X* und N_{after} Bedingungen: • $X \notin N$ • $\forall n \in N_{before}, \forall m \in N_{after}: n, m \in N \wedge m \notin pred^*(n)$
	N_{before}	Menge der Vorgängeraktivitäten	
	N_{after}	Menge der Nachfolgeraktivitäten	
	X	Einzufügende Aktivität	
	template	Aktivitätsvorlage für X	

Bei der Durchführung dieser Änderungsoperation wird zunächst der minimale Kontrollblock bestimmt, der sämtliche Aktivitäten sowohl der Menge N_{before} als auch der Menge N_{after} enthält. Analog zum parallelen Einfügen wird die neue Aktivität *X* dem WSM Net parallel zu dem minimalen Block hinzugefügt; darüber hinaus werden nun jedoch alle Aktivitäten aus N_{before}, die keinen Nachfolger besitzen, der ebenfalls in N_{before} liegt, über eine Synchronisationskante mit *X* verbunden; dasselbe geschieht in analoger Weise mit den ersten Aktivitäten der Menge N_{after}.

Abbildung 81 Auswirkung der ADEPT2 Änderungsoperation »insertBetweenNodeSets«



Neben der bereits existierenden höherwertigen Änderungsoperation »insertBetweenNodeSets« werden an dieser Stelle zwei weitere Änderungsoperationen für ADEPT2 neu definiert. »insertBeforeNodeSet« fügt wie »insertBetweenNodeSets« eine neue Aktivität parallel zu dem minimalen Kontrollblock für zwei Fragmentmengen N_{Before} und N_{After} ein. Allerdings wird das neue Fragment nur mit den Aktivitäten von N_{After} über Synchronisationskanten verbunden, so dass die neue Aktivität vor dem N_{After}-Block, aber parallel zu dem N_{Before}-Block ausgeführt wird. Im Gegensatz dazu sorgt die zweite neue Änderungsoperation »insertAfterNodeSet« dafür, dass nur Synchronisationskanten zwischen der neuen Aktivität und den Aktivitäten von N_{Before}

eingefügt werden. Der Code für »insertBetweenNodeSets« muss nur geringfügig abgewandelt werden, um die neuen Funktionen zu unterstützen; der nachfolgende Abschnitt zeigt den Pseudocode für »insertBeforeNodeSet«:

Formel 17 Pseudocode für die neue höherwertige Änderungsoperation »insertBeforeNodeSet«

```
insertBeforeNodeSet(W, N_before, N_after, X, template) {
      (n_start, n_end) ≔ minBlock(N_before ∪ N_after);
      //Paralleles Einfügen der neuen Aktivität
      parallelInsert(W, X, n_start, n_end, template);
      //Synchronisationskanten einfügen
      FOR (i ≔ 0; i < |N_after| ∧ (∄n ∈ N_after: n ∈ pred*(N_after[i])); i + 1)
         insertSyncEdge(X, N_after[i]);
      END-FOR
}
```

Bei der Ausführung der Funktion werden zunächst der Startknoten und der Endknoten ermittelt, die den minimalen Block begrenzen. Bei »parallelInsert« und »insertSyncEdge« handelt es sich um die bereits beschriebenen, komplexen Änderungsoperationen. Der Pseudocode für »insertAfterNodeSet« unterscheidet sich nur sehr unwesentlich von dem obigen Ausschnitt[43] und wird daher nicht gesondert aufgeführt.

Die nächste Änderungsoperation zum Löschen von Aktivitäten aus ADEPT2 WSM Nets wird nicht im Rahmen der initialen Transformationsphase benötigt. Im Kontext der Rekonfiguration und der erneuten Transformation eines SPF-Graphen ist sie jedoch erforderlich, wenn Knoten aus dem SPF-Graphen entfernt worden sind und diese Modifikationen in Folge auch Auswirkungen auf die Prozessfragmente der Prozessdefinition haben. Aus Gründen der besseren Übersichtlichkeit soll also auch diese Operation zusammen mit den anderen ADEPT2 Änderungsoperationen vorgestellt werden.

Tabelle 23 Entfernen einer Aktivität aus der Prozessdefinition, Change Pattern AP2 [Weber et al. 2008]

Änderungsoperation	Parameter		Semantik
	Name	Typ	
deleteActivity(W, A)	W	WSM Net Prozessdefinition	Löschen einer Aktivität *A* aus der Prozessdefinition *W* Bedingung: • $A \in N - \{n \in N \mid NT(n) \in \{StartFlow, EndFlow\}\}$
	A	Zu entfernende Aktivität	

Die Änderungsoperation »deleteActivity« löscht nicht nur die entsprechende Aktivität aus dem WSM Net, sondern auch sämtliche ihrer lesenden und schreibenden Datenflusskanten sowie die mit ihr verbundenen Synchronisationskanten. Handelt es sich bei *A* um einen *AndSplit-*

[43] In der FOR-Schleife gilt nun die Bedingung: $\nexists n \in N_{before}: n \in succ^*(N_{before}[i])$; die Zeile »insertSyncEdge(X, N_{after}[i])« muss ersetzt werden durch »insertSyncEdge(N_{before}[i], X)«.

Knoten bzw. einen *AndJoin*-Knoten, darf die Aktivität nicht physisch aus dem WSM Net entfernt werden; stattdessen bleibt sie als Nullaktivität in der Prozessdefinition bestehen.

Die letzte Änderungsoperation, die in diesem Kapitel vorgestellt wird, spielt ebenfalls nur bei der Transformation einer Prozessdefinition bzw. einer Prozessinstanz in Folge der Rekonfiguration eine Rolle. Neben dem Löschen einzelner Aktivitäten bietet ADEPT2 auch die Möglichkeit komplette Kontrollblöcke aus der WSM Net Prozessdefinition zu entfernen. Zu diesem Zweck kommt die Änderungsoperation »deleteBlock« zum Einsatz.

Tabelle 24 Entfernen eines vollständigen Kontrollblocks aus der Prozessdefinition, Change Pattern AP2 [Weber et al. 2008]

Änderungsoperation	Parameter		Semantik
	Name	Typ	
deleteBlock(W, A,B)	W	WSM Net Prozessdefinition	Löschen eines Kontrollblocks mit Startaktivität *A* und Endaktivität *B* aus der Prozessdefinition *W* Bedingungen: • $A, B \in W$ • $A \in pred^*(B)$ • $A, B \in N - \{n \in N \mid NT(n) \in \{StartFlow, EndFlow\}\}$ *begrenzen einen Kontrollblock*
	A	Startaktivität des zu entfernenden Kontrollblocks	
	B	Endaktivität des zu entfernenden Kontrollblocks	

»DeleteBlock« löscht nicht nur alle Aktivitäten und Kanten des Kontrollblocks, sondern auch sämtliche lesende und schreibende Datenflusskanten, die Aktivitäten des Kontrollblocks betreffen. Handelt es sich bei *A* und *B* um *AndJoin*-Knoten bzw. *AndSplit*-Knoten, verbleiben sie als Nullaktivitäten in der Prozessdefinition.

4.5.3 Regeln für die Transformation von SPF-Graphen in WSM Nets

Bei der Transformation von SPF-Graphen in WSM Nets gilt es die Strukturierungsmerkmale von SPF-Graphen so zu berücksichtigen, dass strukturell und semantisch sinnvolle Prozessdefinitionen entstehen. In diesem Kapitel wird lediglich die Transformation terminaler Knoten betrachtet. Wie in Abschnitt 4.4.1 definiert wurde, entsprechen SPF-Graphen, die nur noch aus terminalen Knoten bestehen, vollständigen Konfigurationen von SPF-Typ-Graphen. Enthält ein SPF-Graph hingegen noch unkonfigurierte Nichtterminalknoten, so handelt es sich um eine unvollständige Konfiguration; in Kapitel 4.5.5 wird dargestellt, wie sich unvollständige Konfigurationen nutzen lassen, um die Konfiguration zur Laufzeit nachzuholen und die zweite Klasse von Change Pattern, Flexibilität in vordefinierten Regionen, abzubilden.

Bevor aus einem SPF-Graphen eine Prozessdefinition hervorgehen kann, muss eine Entscheidung im Hinblick auf die Knotentypen getroffen werden, die in der Prozessdefinition vorkommen sollen. Prinzipiell unterscheiden sich die terminalen Knoten im SPF-Graphen danach, ob ihre Pendants im SPF-Typ-Graphen Kontextknoten (inklusive der Wurzel als speziellem Kontextknoten) oder SPF-Knoten entsprechen. Da Kontextknoten keine Prozessfragmente als ausführbare Komponenten zugeordnet sind, reduziert sich die mögliche Funktion von Kontextknoten auf Struktur gebende Elemente in WSM Nets; diese Funktion können sie wahrnehmen, indem für jeden Kontextknoten ein Subprozess deklariert wird, der wiederum aus den unter-

geordneten Kontext- und SPF-Knoten besteht. Diese Lösung ist aber wenig zielführend, da Prozessdefinitionen so eine sehr hohe Schachtelungstiefe annehmen können. Während Subprozesse allgemein zur Übersichtlichkeit von Prozessdefinitionen beitragen sollen, würden Subprozesse basierend auf Kontextknoten eher zu überflüssiger Komplexität führen. Ein möglicher Kompromiss besteht darin, nur ausgewählte Kontextknoten als Subprozesse in WSM Nets darzustellen; dies würde allerdings die Erarbeitung von Konzepten für Sichten auf WSM Nets bedeuten, wie z. B. in [Bobrik et al. 2007, Bobrik 2008]. Für diese Arbeit reicht es daher aus, ausschließlich solche terminalen Knoten des SPF-Graphen in WSM Net Aktivitäten zu überführen, die den SPF-Knoten im SPF-Typ-Graphen entsprechen.

Durch diese Entscheidung ergeben sich auch Konsequenzen bei der Übernahme von Knotenattributen durch Aktivitäten im WSM Net. Prinzipiell ist es möglich, dass einer Aktivität sowohl die Attribute des korrespondierenden Knotens im SPF-Graphen als auch die Attribute aller seiner übergeordneten Kontextknoten zugewiesen werden. Da SPF-Typ-Graphen jedoch Überschneidungen zwischen Attributen von Vorgänger- und Nachfolgerknoten nicht ausschließen, könnte dies bei der Übertragung der Attribute auf dieselbe Aktivität zu Problemen führen. Hier könnte Abhilfe geschaffen werden, indem die im SPF-Typ-Graphen vorgegebene Hierarchie Vererbungsrelationen auf Attribute und deren Werte definiert; das bedeutet, ein Nachfolgerknoten im SPF-Typ-Graph würde alle Attribute und Wertzuweisungen seiner übergeordneten Knoten erben und darüber hinaus bei Bedarf um eigene Attribute ergänzen. Eine solche Vererbungshierarchie ist aber in der formalen Spezifikation von SPF-Typ-Graphen bisher nicht vorgesehen und es ist nicht Bestandteil dieser Arbeit, die Sinnhaftigkeit von Vererbungsrelationen auf Attribute zu überprüfen. Deswegen wird im Rahmen dieser Dissertation festgelegt, dass jede Aktivität im WSM Net ausschließlich die Attribute des Knotens im SPF-Graphen erbt, der dem SPF-Knoten im SPF-Typ-Graphen entspricht.

Sei $STG = (V^{STG}, E^{STG}, V^{STG}_{SPF}, r^{STG}, ...)$ ein SPF-Typ-Graph und $SG = (V^{SG}, E^{SG}, V^{SG}_{T}, V^{SG}_{NT}, ...)$ ein SPF-Graph, dann ist die Menge der terminalen Knoten $V_{Trans} \subseteq V^{SG}_{T}$, die während der Transformationsphase von Bedeutung sind, folgendermaßen definiert:

$$V_{Trans} = \{v \in V^{SG}_{T} | \exists u \in V^{STG}_{SPF}: g_v(v) = u\}$$

Bei der Transformation wird für jeden Knoten der Menge V_{Trans} evaluiert, wie das assoziierte Prozessfragment in die WSM Net Prozessdefinition eingefügt werden muss. Die konkrete Art und Weise, in der ein Prozessfragment einer Prozessdefinition hinzugefügt wird, wird von drei wesentlichen Faktoren beeinflusst:

- Enthält die Prozessdefinition außer Start- und Endknoten noch weitere Aktivitäten?
- Steht das aktuell einzufügende Prozessfragment in Beziehung zu anderen Aktivitäten in der Prozessdefinition?
- Gibt es in der Prozessdefinition Aktivitäten, die Klonen des aktuell einzufügenden Prozessfragments entsprechen?

Um den Transformationsvorgang zu erleichtern, könnte es sich bei V_{Trans} um eine geordnete Menge handeln. Dementsprechend würden zunächst alle Prozessfragmente eingefügt, deren Bearbeitung nicht die vorhergehende Ausführung anderer Fragmente bedingt; darüber hinaus würden alle Klone eines Prozessfragments gemeinsam eingefügt. Hierbei ergibt sich jedoch die Schwierigkeit, dass das Herstellen einer solchen geordneten Menge nicht trivial ist, sobald eine Vielzahl von Abhängigkeiten definiert ist. Außerdem kann bei der späteren Rekonfiguration des SPF-Graphen, die wiederum zur Transformation eines Teils des WSM Nets führt, auch nicht von den idealen Bedingungen einer geordneten Menge V_{Trans} ausgegangen werden. Aus diesem Grund wird vorausgesetzt, dass die Elemente der Menge V_{Trans} ungeordnet sind. Das

hat zur Folge, dass vor dem Einfügen jedes Prozessfragments geprüft werden muss, ob das WSM Net bereits Aktivitäten enthält, mit denen das neue Fragment über Constraint-Relationen im SPF-Typ-Graphen in Beziehung steht oder die Klone des neuen Fragments repräsentieren. Wie bei der formalen Spezifikation der Regeln für die Transformation gezeigt wird, werden Klonknoten stets sequentiell hintereinander in die Prozessdefinition eingefügt; d. h. gibt es im Prozess bereits eine Aktivität, die einem Klonknoten entspricht, so dürfen die nächsten Klone nur dahinter angehängt werden. Dies ist unproblematisch, sofern alle Klone gleichartig konfiguriert sind, denn dann entspricht die Abarbeitung der Klonsequenz einem Schleifenkonstrukt in der Prozessdefinition. Sind die Klone jedoch unterschiedlich konfiguriert, muss in den meisten Fällen eine Reihenfolge vorgegeben werden. Das bedeutet, dass für Fachanwender Mittel bereitgestellt werden müssen, die Reihenfolge der zu transformierenden Knoten zu beeinflussen. Da solche Funktionen jedoch unabhängig von dem eigentlichen Transformationsprozess sind, werden sie hier nicht konzeptionell spezifiziert.

Bevor die Regeln für das Einfügen von Prozessfragmenten, die mit den Knoten der Menge V_{Trans} assoziiert sind, aufgestellt werden, werden zunächst einige Mengendefinitionen angeführt. Seien *STG*, *SG* und V_{Trans} wie oben definiert; sei darüber hinaus $W = (N^W, D^W, NT^W, CtrlE^W, ...)$ ein WSM Net und $v_{Trans} \in V_{Trans}$ der Knoten für das aktuell einzufügende Prozessfragment. Für jeden Knoten v_{Trans} müssen die folgenden Mengen ermittelt werden:

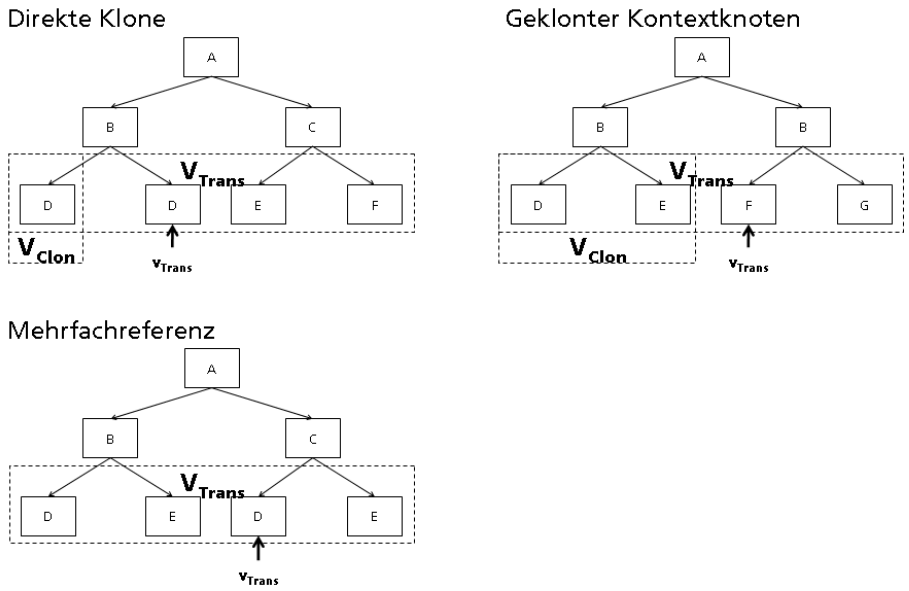
$$V_{Before} = \{v \in V_{Trans} \mid \exists u \in V^{SG}: (u = v \vee u \rightarrow^* v) \wedge \exists z \in V^{SG}: (z = v_{Trans} \vee z \rightarrow^* v_{Trans}) \wedge ((g_v(z), g_v(u), REQUIREMENT) \in C_M \vee (g_v(z), g_v(u), SEQUENCE) \in C_M) \wedge minNode(g_v(z), g_v(u)) = g_v(minNode(z, u))\}$$

$$V_{After} = \{v \in V_{Trans} \mid \exists u \in V^{SG}: (u = v \vee u \rightarrow^* v) \wedge \exists z \in V^{SG}: (z = v_{Trans} \vee z \rightarrow^* v_{Trans}) \wedge ((g_v(u), g_v(z), REQUIREMENT) \in C_M \vee (g_v(u), g_v(z), SEQUENCE) \in C_M) \wedge minNode(g_v(z), g_v(u)) = g_v(minNode(z, u))\}$$

$$V_{Clon} = \{v \in V_{Trans} \mid \exists u \in V^{SG}: (u = v \vee u \rightarrow^* v) \wedge \exists z \in V^{SG}: (z = v_{Trans} \vee z \rightarrow^* v_{Trans}) \wedge u \neq z \wedge g_v(u) = g_v(z) \wedge source(u) = source(z)\}$$

Wie aus den Definitionen hervorgeht, handelt es sich bei V_{Before} um die Menge aller Knoten in V_{Trans}, deren Fragmente vor dem Prozessfragment des Knoten v_{Trans} ausgeführt werden muss. Analog dazu entsprechen die Elemente der Menge V_{After} Knoten, deren Fragmente nach dem Prozessfragment v_{Trans} beendet sein müssen. V_{Clon} enthält alle SPF-Knoten, die Klone von v_{Trans} bzw. einem seiner Vorgänger repräsentieren. Dabei muss wie bei der Identifikation Constraint-relevanter Knoten berücksichtigt werden, dass es Klone zu einem übergeordneten Kontextknoten von v_{Trans} geben kann, so dass diese Menge auch dann Elemente umfasst, wenn kein direkter Klon von v_{Trans} existiert. Klone eines Knoten, die durch Mehrfachreferenzen im SPF-Typ-Graphen entstehen, haben im SPF-Graphen unterschiedliche Vorgänger und sind daher nicht in der Menge mit eingeschlossen. Abbildung 82 zeigt am Beispiel eines SPF-Graphen die verschiedenen Möglichkeiten für die Bildung von V_{Clon}.

Abbildung 82 Mögliche Klon-Konstellationen im SPF-Graphen mit Relevanz für die Transformation



Für jede der Mengen V_{Before}, V_{After} und V_{Clon} muss nun die Menge der entsprechenden Aktivitäten im WSM Net identifiziert werden:

$$N_{Before} = \{n \in N^W | \exists v \in V_{Before}: n = Name^p \cup VID^{SG}(v) \text{ mit } p = VP^{STG}(g_v(v))\}$$

$$N_{After} = \{n \in N^W | \exists v \in V_{After}: n = Name^p \cup VID^{SG}(v) \text{ mit } p = VP^{STG}(g_v(v))\}$$

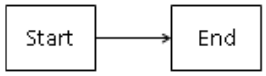
$$N_{Clon} = \{n \in N^W | \exists v \in V_{Clon}: n = Name^p \cup VID^{SG}(v) \text{ mit } p = VP^{STG}(g_v(v))\}$$

Die Bezeichnung einer Aktivität im WSM Net ergibt sich aus dem Namen des Prozessfragments in Verbindung mit der eindeutigen Knotenidentität des entsprechenden Knotens im SPF-Graphen. Das Anfügen der Knotenidentität ist notwendig, da im SPF-Typ-Graphen dasselbe Prozessfragment über die Abbildungsfunktion *VP* von mehreren SPF-Knoten referenziert werden kann; somit verfügt der Name des Prozessfragments allein nicht über die von ADEPT2 geforderte Eindeutigkeit für Aktivitätsbezeichnungen. Die Elemente der Mengen N_{Before} und N_{After} werden im Folgenden auch als »Constraint-relevante Aktivitäten« bezeichnet; bei Elementen der Menge N_{Clon} handelt es sich um so genannte »Klon-Aktivitäten«. Statt der eindeutigen Bezeichnung des Prozessfragments könnte die Verbindung mit einem SPF-Knoten auch über die Implementierung der Aktivität hergestellt werden. Dieses Kriterium würden jedoch alle Aktivitäten erfüllen, die zusammen mit ihrer Implementierung die Spezifikation des Prozessfragments erfüllen, unabhängig davon, ob sie wirklich exakt einem Knoten aus den Mengen V_{Before}, V_{After} oder V_{Clon} entsprechen. Aus diesem Grund ist es bei einer 1:1 Zuordnung zwischen Knoten des SPF-Graphen und Prozessaktivität im WSM Net nicht geeignet.

Initiale Erstellung einer Prozessdefinition

Zu Beginn der Transformation eines SPF-Graphen wird ein initiales WSM Net in ADEPT2 generiert. Jedes WSM Net besteht in Folge der definierten Korrektheitskriterien aus genau einem Startknoten, der über eine gerichtete Kontrollflusskante mit genau einem Endknoten verbunden ist. Diese grundlegende Prozessstruktur (siehe Abbildung 83) wird von ADEPT2 beim Anlegen einer neuen Prozessdefinition automatisch erzeugt.

Abbildung 83 Initiale Struktur eines WSM Net in ADEPT2



Jedes WSM Net ist durch seinen eindeutigen Start- und seinen eindeutigen Endknoten begrenzt. D. h. vor dem Startknoten oder hinter dem Endknoten können keine weiteren Prozessaktivitäten deklariert werden.

Sequentielles Einfügen des ersten Prozessfragments

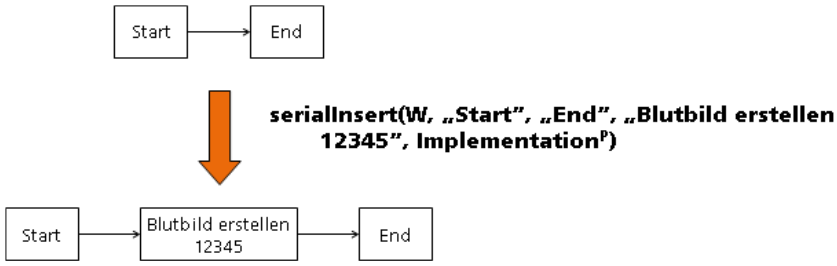
Die Prozessfragmente, für die es im WSM Net weder Constraint-relevante Aktivitäten noch Klon-Aktivitäten gibt, werden grundsätzlich als Teil einer parallelen Verzweigung in die Prozessdefinition eingefügt; auf diese Weise kann in ADEPT2 das Workflow Pattern »Interleaved Parallel Routing« (siehe Kapitel 3.3.1) umgesetzt werden. Da es jedoch aus Gründen der strukturellen Korrektheit nicht erlaubt ist, eine Nullaktivität vom Typ *AndSplit* vor dem Startknoten bzw. eine Nullaktivität vom Typ *AndJoin* hinter dem Endknoten des WSM Net einzufügen, muss die erste Aktivität sequentiell zwischen Start- und Endknoten angeordnet werden. Es gelten die oben angeführten Mengendefinitionen; dann lautet die Bedingung für das sequentielle Einfügen des ersten Prozessfragments:

$N^W - \{n \in N^W \mid NT^W(n) = StartFlow \vee NT^W(n) = EndFlow\} = \varnothing$

(Bedingung für die Nicht-Existenz von Aktivitäten)

Die nächste Abbildung demonstriert den Einfügevorgang an einem Beispiel.

Abbildung 84 Sequentielles Einfügen des ersten Prozessfragments



Bei $Implementation^P$ handelt es sich um die Implementierung der Schnittstellenspezifikation des neu eingefügten Prozessfragments. Damit Fragmentimplementierungen in ADEPT2 genutzt werden können, müssen sie zuvor als Aktivitätsvorlagen in das ADEPT2 Activity Repository eingestellt werden.

Paralleles Einfügen unabhängiger Prozessfragmente

Die Nicht-Existenz von Constraint-Relationen vom Typ Anforderung und Sequenz impliziert, dass das mit dem Abbild des aktuellen Knoten v_{Trans} im SPF-Typ-Graphen assoziierte Prozessfragment in keiner vorgegebenen Reihenfolge zu anderen Fragmenten ausgeführt werden muss, sondern der Zeitpunkt der Ausführung zur Laufzeit frei gewählt werden kann. Dies gilt auch für solche Prozessfragmente, für die im WSM Net bisher weder Constraint-relevante Aktivitäten noch Klon-Aktivitäten existieren. Nach dem sequentiellen Hinzufügen des ersten Prozessfragments, werden derartige Fragmente daher standardmäßig parallel zu den bereits

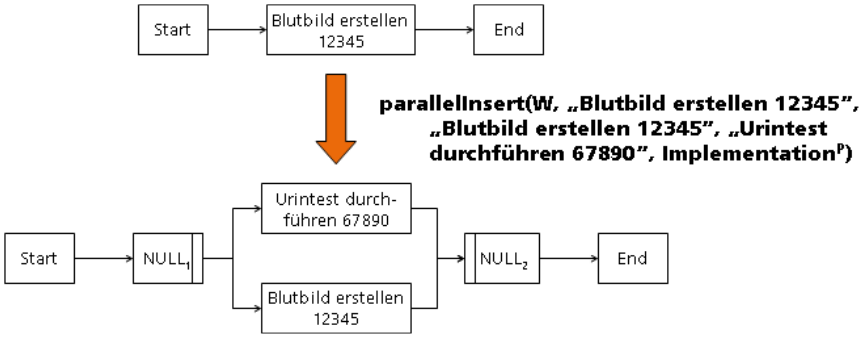
vorhandenen Aktivitäten in die Prozessdefinition eingefügt. Für das parallele Einfügen eines Prozessfragments in *W* müssen die folgenden Bedingungen erfüllt sein:

$N^W - \{n \in N^W \mid NT^W(n) = StartFlow \vee NT^W(n) = EndFlow\} \neq \varnothing$
(Bedingung für die Existenz von Aktivitäten)

$N_{Before} = N_{After} = N_{Clon} = \varnothing$
(Bedingung für die Unabhängigkeit des Prozessfragments)

Treffen beide Bedingungen zu, darf das Prozessfragment parallel zu allen im WSM Net bereits vorhandenen Aktivitäten eingefügt werden (siehe Abbildung 85).

Abbildung 85 Paralleles Einfügen unabhängiger Prozessfragmente



Einfügen von Prozessfragmenten unter Berücksichtigung von Constraint-relevanten Aktivitäten und Klon-Aktivitäten

Lediglich die Constraints »Anforderung« und »Sequenz« wirken sich auf die Struktur von WSM Nets aus; dabei ist die Art und Weise, in der sie den Aufbau der Prozessdefinition beeinflussen, bei beiden Aktivitäten gleich. Zum Zeitpunkt der Transformation muss davon ausgegangen werden, dass sämtliche Constraints in der Struktur des SPF-Graphen berücksichtigt sind. Existiert für den aktuellen Knoten $v_{Trans} \in V_{Trans}$ ein Anforderungs- oder Sequenzbedingung zu einem anderen Knoten im SPF-Typ-Graphen, so darf das neue Fragment nur vor bzw. hinter den Aktivitäten eingefügt werden, zu denen die Abhängigkeitsbeziehung besteht. Da es sich bei V_{Trans} nicht um eine geordnete Menge handelt, kann zum Zeitpunkt des Einfügens des neuen Prozessfragments der Fall auftreten, dass die Constraint-relevanten Aktivitäten, deren Bearbeitung vor dem Fragment erfolgen soll, bereits teilweise oder komplett im WSM Net enthalten sind; dasselbe gilt für Constraint-relevante Aktivitäten, die erst nach Beendigung des Fragments ausgeführt werden dürfen. Die Art des Einfügens des neuen Fragments erfolgt dann in Abhängigkeit der Position und des Typs der Constraint-relevanten Aktivitäten. Für Klon-Aktivitäten gilt, dass diese genauso wie Aktivitäten der Menge N_{Before} behandelt werden; d. h. das neue Prozessfragment darf erst nach Beendigung der Klon-Aktivitäten ausgeführt werden. Die Tatsache, dass jeder Klon nach den bereits existierenden Klon-Aktivitäten in die Prozessdefinition eingefügt wird, hat zur Folge, dass sich mit Hilfe von Klonen statische Prozessschleifen im WSM Net realisieren lassen. Diese sind statisch, da die Anzahl der Klone im SPF-Graphen zur Modellierungszeit determiniert wird. Erst mit der Rekonfiguration lassen sich auf diese Weise auch dynamische Schleifen umsetzen. Die Bedingungen für das Einfügen eines Prozessfragments unter Berücksichtigung von Constraint-relevanten Aktivitäten und Klon-Aktivitäten sind folgendermaßen spezifiziert:

$N^W - \{n \in N^W \mid NT^W(n) = StartFlow \vee NT^W(n) = EndFlow\} \neq \varnothing$

(Bedingung für die Existenz von Aktivitäten)

$N_{Before} \neq \varnothing \vee N_{After} \neq \varnothing$ *mit* $N_{Before} := N_{Before} \cup N_{Clon}$
(Bedingung für die Existenz Constraint-relevanter Aktivitäten und/oder Klon-Aktivitäten)

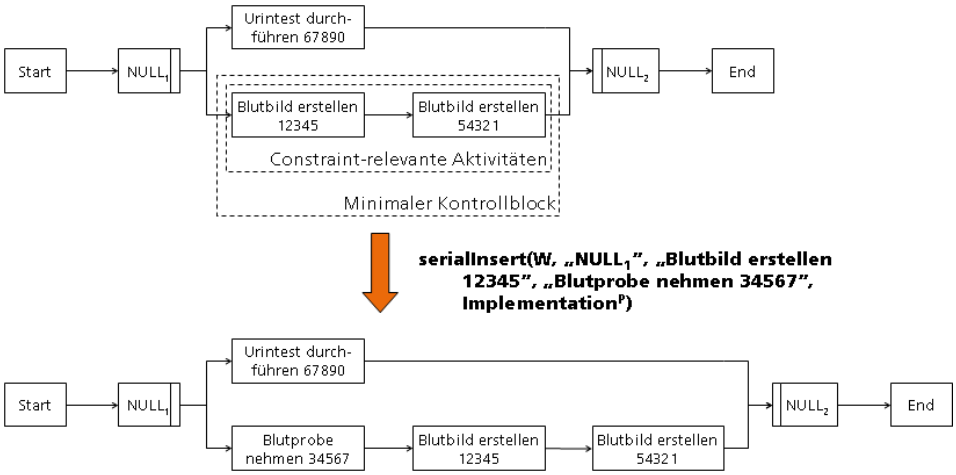
Eine einfache und generische Möglichkeit ein deklaratives Modell wie SPF-Graphen in ein prozedurales Modell wie WSM Nets zu übertragen, besteht im Anlegen von parallelen Verzweigungen und der Verknüpfung von abhängigen Aktivitäten über die Synchronisationskanten in ADEPT2. Dies führt jedoch auch zu extrem unübersichtlichen Prozessdefinitionen, die, trotz der korrekten Ausführung durch das WfMS, für Menschen kaum noch verständlich sind. Auch wenn das WSM Net als resultierende Prozessdefinition nicht für die Darstellung für fachliche Anwender geeignet ist, kann es dennoch als Verständigungsgrundlage für technische Mitarbeiterinnen und Mitarbeiter dienen. Das Vorliegen von Constraint-Relationen und Klon-Aktivitäten sollte daher ausgenutzt werden, um die Struktur des WSM Net möglichst stark zu sequentialisieren. Um festzustellen, in welcher Weise das neue Prozessfragment in das WSM Net eingefügt werden kann, muss zunächst der minimale Kontrollblock über die Aktivitätenmengen $N_{Before} \cup N_{Clon}$ und N_{After} ermittelt werden. Überschneidungen zwischen diesen Blöcken sind möglich, allerdings darf keine Konstellation auftreten, in der eine Aktivität aus N_{After} vor einer Aktivität von $N_{Before} \cup N_{Clon}$ ausgeführt wird. Die Entstehung solcher Fehlerquellen ist bei Einhaltung der Korrektheitskriterien für SPF-Typ-Graphen, die in Kapitel 4.3.6 beschrieben wurden, ausgeschlossen.

Betrachten wir als erstes folgende Situation: $N_{After} \neq \varnothing$ und $N_{Before} \cup N_{Clon} = \varnothing$; d. h. es gibt im WSM Net Constraint-relevante Aktivitäten, die nach dem neuen Prozessfragment ausgeführt werden müssen. Sei $(start_{min}, end_{min}) = minBlock(N_{After})$ die Start- bzw. Endaktivität des minimalen Blocks über alle Constraint-relevanten Aktivitäten. Folgende Bedingung muss für das sequentielle Einfügen des neuen Prozessfragments erfüllt sein:

$(start_{min} \in N_{After} \vee succ(start_{min}) \subseteq N_{After}) \wedge (\nexists\ m \in pred^*(start_{min}): NT^W(m) \notin \{StartFlow, AndSplit\}) \Rightarrow serialInsert(W, pred(start_{min}), start_{min}, X, template)$

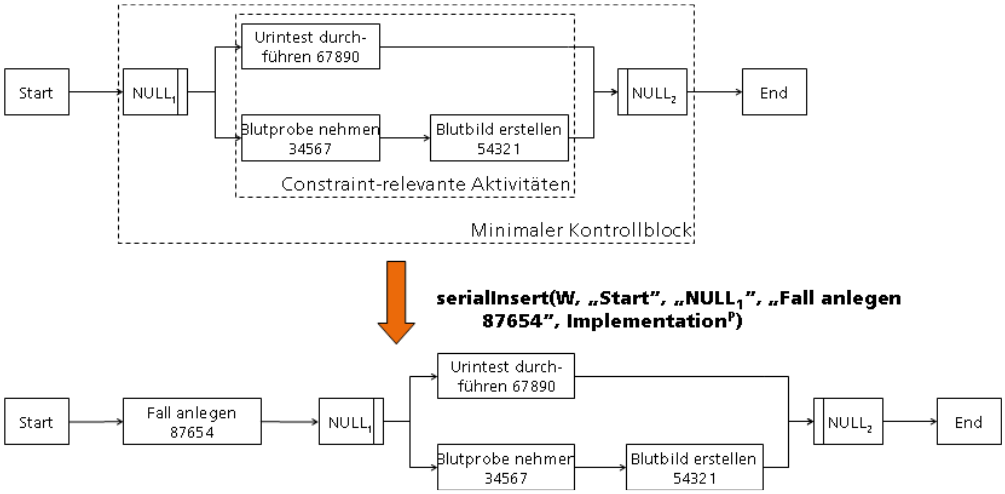
Die folgende Abbildung verdeutlicht das sequentielle Einfügen des neuen Prozessfragments, wenn die obigen Bedingungen erfüllt sind.

Abbildung 86 Sequentielles Einfügen eines Prozessfragments vor Constraint-relevanten Aktivitäten (Beispiel 1)



Angenommen der SPF-Typ-Graph enthält einen Sequenz-Constraint, dem zufolge Prozessfragmente vom Typ »Blutbild erstellen« erst nach dem Fragment »Blutprobe nehmen« bearbeitet werden. Als Constraint-relevante Aktivitäten im WSM Net werden daher die beiden Klon-Aktivitäten für »Blutbild erstellen« identifiziert. Die Feststellung des minimalen Blocks über die Klon-Aktivitäten ergibt, dass der Startknoten des Blocks einer Constraint-relevanten Aktivität entspricht. Darüber hinaus gibt es außer dem *AndSplit*-Knoten $NULL_1$ keine Vorgängeraktivitäten. Infolgedessen ist das sequentielle Einfügen des neuen Prozessfragments vor den beiden Klon-Aktivitäten möglich. Ein anderes Beispiel für das sequentielle Einfügen unter Berücksichtigung einer Constraint-Relation zeigt Abbildung 87.

Abbildung 87 Sequentielles Einfügen eines Prozessfragments vor Constraint-relevanten Aktivitäten (Beispiel 2)



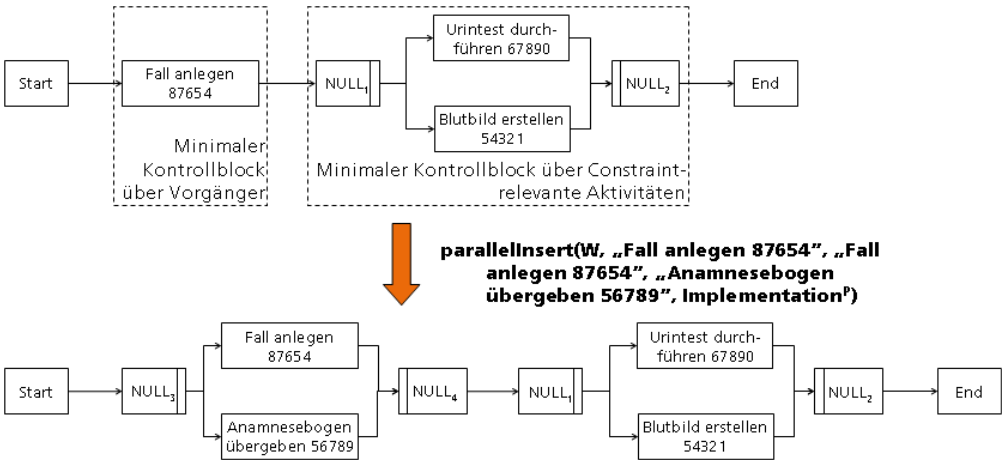
Im SPF-Typ-Graphen gibt es eine Anforderungsbedingung, die besagt, dass zunächst ein Fall angelegt werden muss, bevor ein Urintest durchgeführt, eine Blutprobe genommen und ein Blutbild erstellt werden kann. $NULL_1$ entspricht nun dem Startknoten des minimalen Blocks über die Constraint-relevanten Aktivitäten. Da es vor $NULL_1$ allerdings keine Vorgängeraktivitäten im WSM Net gibt, kann das neue Fragment sequentiell zwischen dem Startknoten der Prozessdefinition und dem Startknoten des minimalen Kontrollblocks eingefügt werden.

Sind die Bedingungen für das sequentielle Einfügen des abhängigen Prozessfragments nicht erfüllt, so wird als nächstes der minimale Block der unabhängigen Vorgängeraktivitäten abzüglich von Start- und *AndSplit*-Knoten im WSM Net ermittelt. Sei also $(start^*_{min}, end^*_{min}) = minBlock(pred^*(N_{After}) - \{n \in N^W | NT^W(n) \notin \{StartFlow, AndSplit\}\})$, dann gilt:

$$end^*_{min} \in pred(start_{min}) \Rightarrow parallelInsert(W, start^*_{min}, end^*_{min}, X, template)$$

Dieser Bedingung zufolge gibt es zwischen dem minimalen Block über die Vorgängerknoten der Constraint-relevanten Aktivitäten endend bei Knoten end^*_{min} und ihrem minimalen Block beginnend mit Knoten $start_{min}$ keine Überschneidungen; das neue Prozessfragment muss also nur parallel zum minimalen Block der Vorgängeraktivitäten eingefügt werden (siehe Abbildung 88).

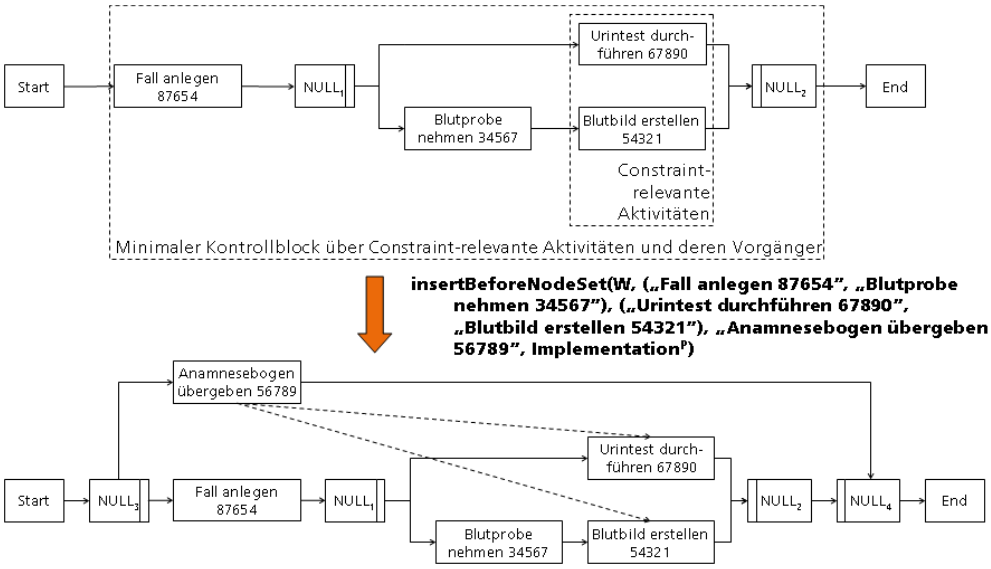
Abbildung 88 Einfügen eines Prozessfragments parallel zu den Vorgängern der Constraint-relevanten Aktivitäten



In dem Beispiel soll das Prozessfragment »Anamnesebogen übergeben 56789« in die Prozessdefinition eingefügt werden, wobei im SPF-Typ-Graphen definiert ist, dass dieses Fragment vor den Fragmenten für »Urintest durchführen 67890« und »Blutbild erstellen 54321« erfolgen muss. Das sequentielle Einfügen ist nicht möglich, da zu dem Prozessfragment »Fall anlegen 87654« keine solche Constraint-Bedingung spezifiziert ist. Folglich wird das neue Fragment parallel zu der Aktivität »Fall anlegen 87654« eingefügt, bei der es sich um die Vorgängeraktivität der beiden Constraint-relevanten Aktivitäten handelt.

Ist die Schnittmenge zwischen dem minimalen Kontrollblock über die Constraint-relevanten Aktivitäten und dem minimalen Kontrollblock über ihre Vorgängeraktivitäten nicht leer, so bleibt nur die Möglichkeit, das neue Prozessfragment parallel zu der Menge der Vorgänger und der Menge der Constraint-relevanten Aktivitäten einzufügen und die Abhängigkeiten durch Synchronisationskanten festzulegen. Wie in der folgenden Abbildung gezeigt wird, wird dazu die semantisch hochwertige Änderungsoperation *insertBeforeNodeSet* benötigt. Gemäß dem Algorithmus für diese Änderungsoperation aus Kapitel 4.5.2 werden die Synchronisationskanten nur zu denjenigen Constraint-relevanten Aktivitäten gezogen, die keine Vorgänger in der Menge N_{After} haben.

Abbildung 89 Einfügen eines Prozessfragments parallel zu Constraint-relevanten Aktivitäten und deren Vorgänger

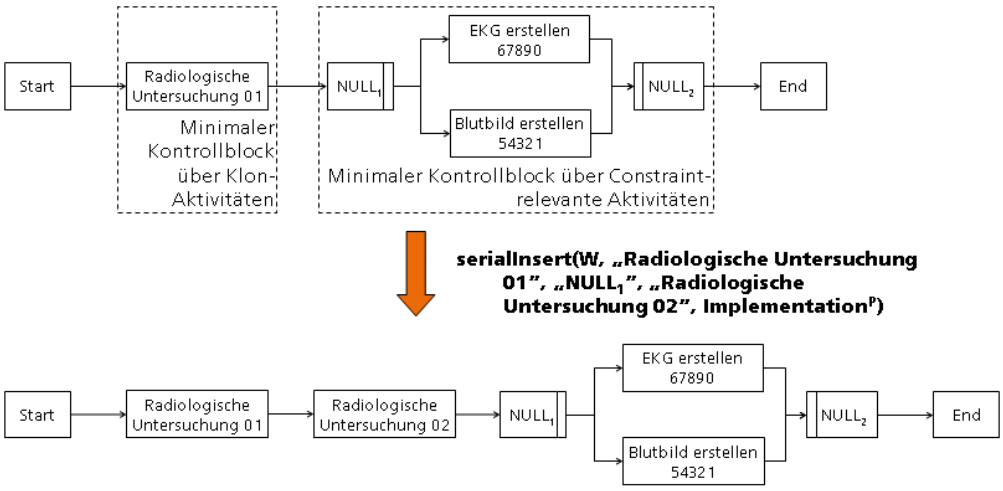


In dem oben dargestellten Beispiel besteht keine Abhängigkeit zwischen dem Prozessfragment »Anamnesebogen übergeben 56789« und den Fragmenten »Fall anlegen 87654« und »Blutprobe nehmen 34567«; bei der Ermittlung der minimalen Kontrollblöcke existiert daher eine Überscheidung. Folglich kann das neue Fragment nur parallel zu beiden Kontrollblöcken eingefügt werden. Um dennoch eine Reihenfolge zwischen der zusätzlichen Aktivität und den Constraint-relevanten Aktivitäten vorzugeben, wird die Ablaufstruktur um entsprechende Synchronisationskanten ergänzt.

Für den Fall $N_{After} = \varnothing$ und $N_{Before} \cup N_{Clon} \neq \varnothing$ lassen sich die Bedingungen und Einfügevorgänge in analoger Weise spezifizieren, so dass eine gesonderte Betrachtung nicht notwendig ist. Aus diesem Grund soll abschließend die Situation untersucht werden, in der das WSM Net sowohl Aktivitäten enthält, die vor dem neu einzufügenden Prozessfragment bearbeitet werden müssen, als auch Aktivitäten, die erst im Anschluss daran durchgeführt werden dürfen; es gilt also: $N_{After} \neq \varnothing$ und $N_{Before} \cup N_{Clon} \neq \varnothing$. Das sequentielle Einfügen des neuen Prozessfragments ist grundsätzlich möglich, wenn sich die minimalen Kontrollblöcke der Aktivitätenmengen $N_{Before} \cup N_{Clon}$ und N_{After} nicht überschneiden und es zwischen zwei Elementen aus $N_{Before} \cup N_{Clon}$ und N_{After} keine anderen Elemente gibt. Seien $(start^{Before}_{min}, end^{Before}_{min}) = minBlock(N_{Before} \cup N_{Clon})$ und $(start^{After}_{min}, end^{After}_{min}) = minBlock(N_{After})$, dann gilt:

$$(end^{Before}_{min} \in N_{Before} \vee pred(end^{Before}_{min}) \subseteq N_{Before}) \wedge (start^{After}_{min} \in N_{After} \vee succ(start^{After}_{min}) \subseteq N_{After}) \wedge start^{After}_{min} \in succ^*(end^{Before}_{min}) \wedge (\nexists\, m \in succ^*(end^{Before}_{min}) \cap pred^*(start^{After}_{min})) \Rightarrow serialInsert(W, end^{Before}_{min}, start^{After}_{min}, X, template)$$

Abbildung 90 Serielles Einfügen eines neuen Prozessfragments zwischen zwei Aktivitätsmengen



In dem obigen Beispiel sollen zwei radiologische Untersuchungen durchgeführt werden, wobei die Art der Untersuchung nur durch die Attributierung festgelegt wurde. Damit existiert im WSM Net bereits eine Klonaktivität, die der Menge N_{Before} hinzugefügt wird. Da das EKG und das Blutbild erst nach der radiologischen Untersuchung erstellt werden sollen, existieren jedoch auch zwei Constraint-relevante Aktivitäten in der Menge N_{After}. Weil sich beide Mengen nicht überschneiden und auch keine unabhängigen Aktivitäten sequentiell zwischen oder parallel zu den Mengen ausgeführt werden, ist es möglich, das neue Fragment sequentiell einzufügen.

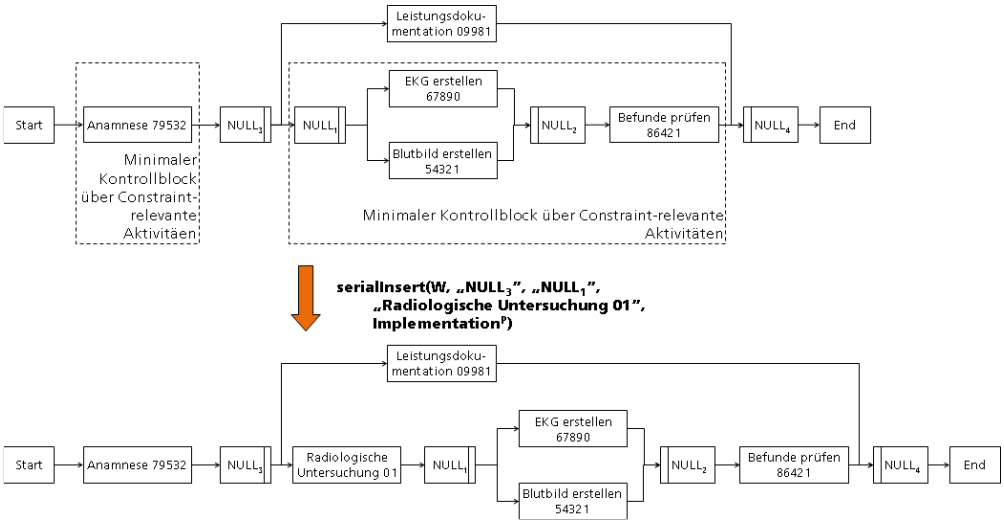
Das sequentielle Einfügen ist auch dann möglich, wenn es zwischen den beiden minimalen Kontrollblöcken Aktivitäten gibt, die entweder nur vom Typ *AndJoin* oder nur vom Typ *AndSplit* sind:

$$(end^{Before}_{min} \in N_{Before} \vee pred(end^{Before}_{min}) \subseteq N_{Before}) \wedge (start^{After}_{min} \in N_{After} \vee succ(start^{After}_{min}) \subseteq N_{After}) \wedge start^{After}_{min} \in succ^*(end^{Before}_{min}) \wedge (\forall m \in succ^*(end^{Before}_{min}) \cap pred^*(start^{After}_{min}): NT^W(m) = AndJoin) \Rightarrow serialInsert(W, end^{Before}_{min}, succ(end^{Before}_{min}), X, template)$$

$$(end^{Before}_{min} \in N_{Before} \vee pred(end^{Before}_{min}) \subseteq N_{Before}) \wedge (start^{After}_{min} \in N_{After} \vee succ(start^{After}_{min}) \subseteq N_{After}) \wedge start^{After}_{min} \in succ^*(end^{Before}_{min}) \wedge (\forall m \in succ^*(end^{Before}_{min}) \cap pred^*(start^{After}_{min}): NT^W(m) = AndSplit) \Rightarrow serialInsert(W, pred(start^{After}_{min}), start^{After}_{min}, X, template)$$

In dem nächsten Beispiel wird ebenfalls ein Fragment zur radiologischen Untersuchung in ein WSM Net eingefügt. Es bestehen Abhängigkeiten zur Anamnese, die vorher durchgeführt werden muss, und zu den Aktivitäten »EGK erstellen 67890«, »Blutbild erstellen 54321« und »Befunde prüfen 86421«, die erst im Anschluss erfolgen dürfen; parallel zu den drei letztgenannten Aktivitäten findet jedoch die Aktivität »Leistungsdokumentation 09981« statt. Das bedeutet, dass das neue Prozessfragment sequentiell zwischen die *AndSplit*-Knoten »$NULL_3$« und »$NULL_1$« eingefügt werden kann (siehe Abbildung 91).

Abbildung 91 Sequentielles Einfügen eines Prozessfragments zwischen zwei Aktivitätenmengen unter Berücksichtigung unabhängiger Aktivitäten



Ist das sequentielle Einfügen nicht möglich, weil zwischen den ersten Aktivitäten aus N_{Before} und den letzten Aktivitäten aus N_{After} unabhängige Aktivitäten liegen, so muss überprüft werden, ob das neue Fragment parallel dazu eingefügt werden kann. Dazu muss zunächst der minimale Kontrollblock über die Schnittmenge der Nachfolger von end^{Before}_{min} und der Vorgänger von $start^{After}_{min}$ gebildet werden:

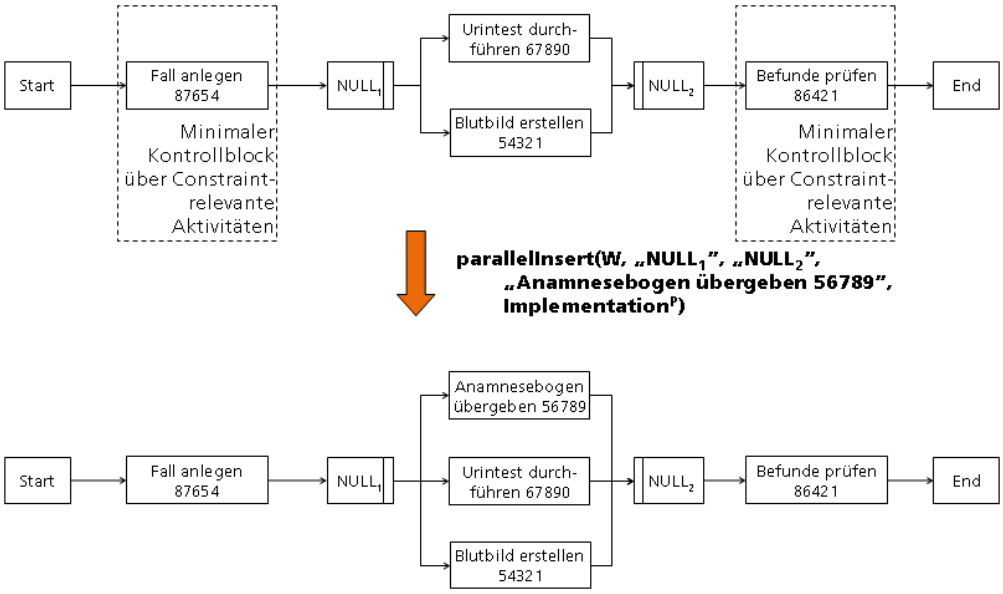
$$(start^*_{min}, end^*_{min}) = minBlock(succ^*(end^{Before}_{min}) \cap pred^*(start^{After}_{min}))$$

Liegt dieser minimale Block nun exakt zwischen den Kontrollblöcken über N^{Before} und N^{After}, dann kann das neue Prozessfragment parallel zu diesem Block in die Prozessdefinition eingefügt werden:

$$start^*_{min} \in succ(end^{Before}_{min}) \wedge end^*_{min} \in pred(start^{After}_{min}) \Rightarrow parallelInsert(W, start^*_{min}, end^*_{min}, X, template)$$

Die nächste Abbildung illustriert den Einfügevorgang für das Prozessfragment »Anamnesebogen übergeben 56789«. Vor der Übergabe muss zuerst der Fall angelegt werden und die Anamnesedaten müssen in die Aktivität zur Prüfung der Befunde einfließen. Von den Aktivitäten »Urintest durchführen 67890« und »Blutbild erstellen 54321« ist das neue Fragment allerdings unabhängig; dementsprechend kann es parallel zu diesen beiden Aktivitäten in das WSM Net eingefügt werden.

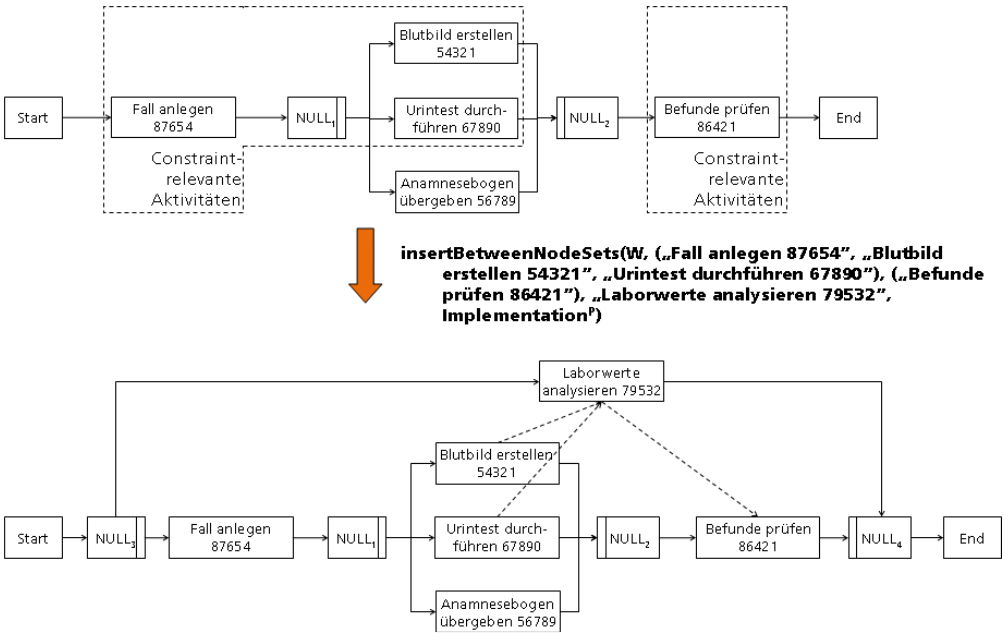
Abbildung 92 Einfügen eines Prozessfragments zwischen zwei Aktivitätenmenge und parallel zu unabhängigen Aktivitäten



Parallel darf nur eingefügt werden, wenn zu jedem *AndSplit*-Knoten der Schnittmenge ein entsprechender *AndJoin*-Knoten existiert und umgekehrt bzw. wenn es sich bei allen Aktivitäten der Schnittmenge um sequentiell aufeinanderfolgende Aktivitäten handelt. Wenn diese beiden Bedingungen nicht erfüllt sind, z. B. weil die Aktivitäten von N_{Before} und N_{After} in zueinander parallelen Zweigen vorkommen und sich ihre minimalen Kontrollblöcke überlappen, so bleibt nur die Möglichkeit, das neue Prozessfragment parallel zu den minimalen Blöcken einzufügen und über Synchronisationskanten mit den Constraint-relevanten Aktivitäten zu verbinden; dies geschieht durch Nutzung der hochwertigen Änderungsoperation *insertBetweenNodeSets*.

In dem folgenden Beispiel soll ein Fragment zur Analyse der Laborwerte in das WSM Net eingefügt werden. Dieses Fragment kann nur dann bearbeitet werden, wenn ein Fall angelegt sowie ein Blutbild erstellt und ein Urintest durchgeführt wurde. Außerdem muss die Ausführung der neuen Aktivität beendet sein, bevor die Befunde überprüft werden.

Abbildung 93 Einfügen eines Prozessfragments zwischen zwei Aktivitätenmengen



Gemäß der Abbildung ergibt sich zwischen den Mengen N_{Before} und N_{After} zwar keine Schnittmenge, jedoch gibt es eine unabhängige Aktivität, die parallel zur Analyse der Laborwerte stattfinden kann. Aus diesem Grund sind das sequentielle und auch das einfache parallele Einfügen nicht möglich.

Attributierung der eingefügten Prozessfragmente

Nach dem Einfügen eines neuen Prozessfragments in ein WSM Net werden abschließend die Attribute von v_{Trans} auf die Aktivität übertragen. Dies erfolgt durch Nutzung der bereits vorgestellten Änderungsoperation »changeActivityAttribute«:

$$\forall a \in A^{SG}: VA^{SG}(v_{Trans}, a) \neq Undefined \Rightarrow changeActivityAttribute(W, Name^{p} \cup VID^{SG}(v_{Trans}), a, VA^{SG}(v_{Trans}, a)) \text{ mit } p = VP^{STG}(g_{v}(v_{Trans}))$$

4.5.4 Ergänzung des Datenflusses auf Basis der Schnittstellenspezifikationen der Prozessfragmente

Wie in Kapitel 4.3.4 festgelegt wurde, umfasst die Schnittstellenspezifikation jedes Prozessfragments Angaben zu den Datenobjekten, die von dem Fragment konsumiert und produziert werden. Dazu gehört die Information, von welchem Datentyp das Objekt ist und ob es sich um ein obligatorisches oder optionales Datenobjekt handelt. Sobald also zu einem Knoten v_{Trans} aus der Menge V_{Trans} das zugehörige Prozessfragment identifiziert wurde, steht implizit fest, welche Datenobjekte das Fragment zur Ausführung benötigt und welche Objekte nach Beendigung des Fragments im Datenkontext des WfMS zur Verfügung stehen. Beim Einfügen eines neuen Prozessfragments in das WSM Net lassen sich also zunächst die mit dem Fragment assoziierten Datenobjekte automatisch erzeugen. Jedes Prozessfragment p umfasst eine Menge von Methoden $m \in Methods^{p}$, die wiederum mit einer Menge von Parametern $Parameters^{m}$ verknüpft sind. Ob für einen Parameter ein neues Datenobjekt erzeugt und der Men-

ge der Datenobjekte *D* eines WSM Nets $W = (N^W, D^W, NT^W, ...)$ hinzugefügt wird, hängt davon ab, ob es dort bereits existiert.

$$\forall param \in Parameters^m \nexists d \in D^W: (Name^{param} \cup param) = d \Rightarrow addDataElements(W, Name^{param} \cup param, DataType^{param}, NULL)$$

Bei dem Abgleich des neuen Parameters mit den bereits vorhandenen Datenobjekten genügt die Angabe der eindeutigen Parameterbezeichnung. Diese setzt sich zum einen aus dem Attribut $Name^{param}$ und zum anderen aus der Identität des jeweiligen Parameters zusammen. Da gemäß Spezifikation von Prozessfragmenten jeder Parameter eine eigene Entität bildet, die in verschiedenen Methoden beliebiger Prozessfragmente wiederverwendet werden kann (siehe Kapitel 4.3.4), muss der Datentyp übereinstimmen, wenn die eindeutige Parameterbezeichnung identisch ist. Würde als Vergleichskriterium nur der Datentyp genannt, wäre die Wahrscheinlichkeit für unerwünschte Datenflüsse hingegen sehr hoch. Die Nutzung der eindeutigen Parameterbezeichnung bei der Transformation hat bei der Spezifikation von Prozessfragmenten zur Folge, dass überprüft werden muss, in welchen Beziehungen das Fragment zu anderen Fragmenten stehen soll, damit bei Bedarf in den Methoden dieselben Parameter verwendet werden. Diese Kontrollen sollten in einem iterativen Prozess nach der Modellierung des SPF-Typ-Graphen stattfinden, da in diesem Aufschluss über Zusammenhänge zwischen Prozessfragmenten gegeben werden.

Die Funktion »addDataElements« repräsentiert ein Änderungsprimitiv in ADEPT2, das das Hinzufügen von Datenelementen zu WSM Nets bewirkt. Als Parameter wird der Funktion das jeweilige WSM Net, die Bezeichnungen der einzufügenden Datenobjekte, deren Wertebereich und Default-Wert übergeben. Nachdem alle Datenobjekte in die Prozessdefinition eingefügt wurden, müssen noch die lesenden und schreibenden Datenflusskanten zu den Aktivitäten hin ergänzt werden.

$$\forall param \in Parameter^m: IOType^{param} \in \{IN, IN/OUT\} \Rightarrow addDataEdges(W, (Name^p \cup VID^{SG}(v_{Trans}), Name^{param} \cup param, read))$$

$$\forall param \in Parameter^m: IOType^{param} \in \{OUT, IN/OUT\} \Rightarrow addDataEdges(W, (Name^p \cup VID^{SG}(v_{Trans}), Name^{param} \cup param, write))$$

Die Methode »addDataEdges« zählt ebenfalls zu den Änderungsprimitiven in ADEPT2 und bewirkt das Hinzufügen einer lesenden oder schreibenden Datenflusskante. Als Parameter erhält sie das WSM Net und ein Objekt vom Typ *DataE*. Wird ein Prozessfragment in das WSM Net eingefügt, für dessen Methodenparameter bereits Datenobjekte existieren, so werden lediglich Datenflusskanten zwischen der entsprechenden Aktivität und dem Datenobjekt erzeugt.

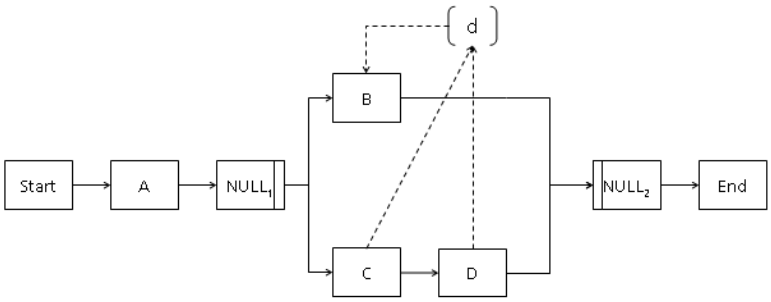
Während die Integrität des Kontrollflusses durch Überprüfung der Einhaltung der für SPF-Typ-Graphen definierten Korrektheitskriterien und die Befolgung der Transformationsregeln sichergestellt werden kann, wurden für die Integrität des Datenflusses bisher keine solchen Kriterien definiert; Datenabhängigkeiten werden im SPF-Typ-Graphen nicht explizit festgelegt (siehe Kapitel 4.3.3). Dies ist auch sinnvoll, da der Umgang mit Datenflüssen von Modellierungssprache zu Modellierungssprache und WfMS zu WfMS unterschiedlich ist, so dass auf Seiten des SPF-Typ-Graphen keine Regelungen getroffen werden sollten, die die Gestaltung des Datenflusses unnötig einschränken würden. Das automatische Einfügen von Datenobjekten und Datenflusskanten bedeutet jedoch nicht, dass auch die Integrität der Datenflüsse automatisch gesichert ist. Dies hat die Konsequenz, dass der Autor des SPF-Typ-Graphen die Verantwortung trägt, dass aus der Transformation vollständig konfigurierter SPF-Typ-Graphen aus-

schließlich solche Prozessdefinitionen hervorgehen können, die im Hinblick auf ihre Datenflüsse korrekt sind. Wichtig ist daher, dass die Mittel zur Verfügung stehen, um Prozessfragmente durch die Knotenstruktur des SPF-Typ-Graphen so miteinander in Beziehung zu setzen, dass die Anforderungen des ausführenden WfMS an die Modellierung der Datenflüsse eingehalten werden.

Diese Mittel sind durch die Bereitstellung von Constraint-Relationen gegeben. Wird von einem Prozessfragment ein Datenobjekt produziert, das von einem anderen Fragment benötigt wird, so sollte ein Constraint zwischen den entsprechenden Knoten im SPF-Typ-Graphen definiert werden. Handelt es sich darüber hinaus um ein obligatorisches Datenobjekt, bietet sich die Nutzung des Constraints »Anforderung« an; ist das Objekt hingegen optional, ist keine Constraint-Relation erforderlich oder es genügt ein Constraint vom Typ »Sequenz«. Ausschluss-Constraints machen im Hinblick auf den Datenfluss in WSM Nets z. B. dann Sinn, wenn durch das Einfügen von zwei Prozessfragmenten konkurrierende Schreibzugriffe auf dasselbe Datenobjekt stattfinden würden.

Da sich die Korrektheitsanforderungen in ADEPT2 auch auf den Datenfluss erstrecken, gelten rigide Vorschriften, die bei der Definition des Datenflusses einzuhalten sind. Eine Bedingung ist, dass eine Aktivität kein Datenobjekt konsumieren darf, das nicht vorher durch eine andere Aktivität erzeugt worden ist. Eine zweite Bedingung schließt konkurrierende Schreib-/Lesezugriffe auf Datenobjekte durch Aktivitäten paralleler Zweige aus. Diese Vorgaben müssen bei der Spezifikation des SPF-Typ-Graphen unbedingt eingehalten werden, um konsistente WSM Nets erzeugen zu können. Die Einhaltung der ersten Bedingung wird dadurch erleichtert, dass aus der Transformation keine WSM Nets hervorgehen können, die bedingte Verzweigungen oder Schleifenkonstrukte enthalten. Der deklarative Modellcharakter von SPF-Graphen erschwert jedoch die Erfüllung der zweiten Bedingung. Abbildung 94 zeigt ein Beispiel für ein WSM Net, das aus mehreren Gründen von ADEPT2 als ungültig deklariert werden würde.

Abbildung 94 Beispiel für ein WSM Net mit inkonsistentem Datenfluss

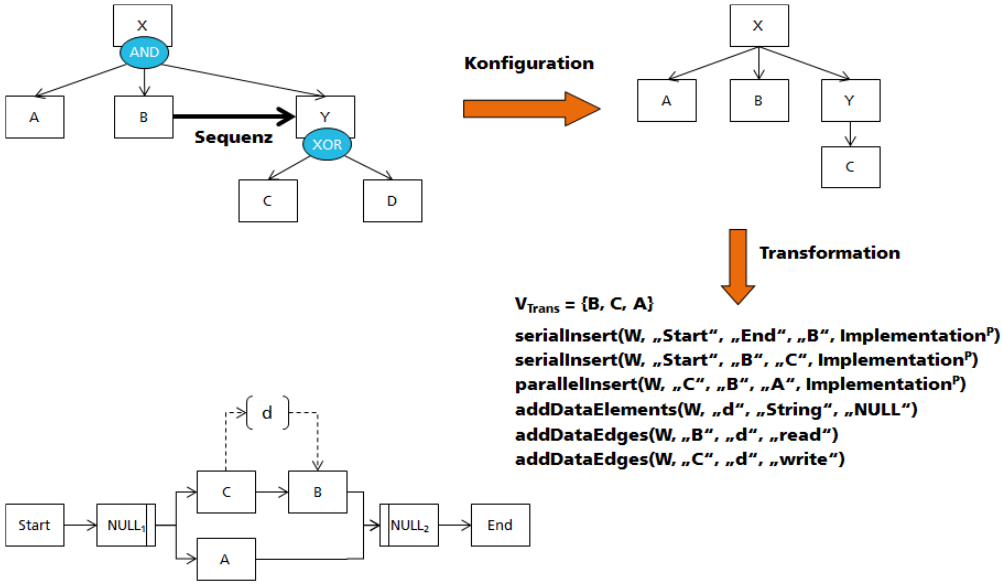


Da die Knoten *B*, *C* und *D* Teil einer parallelen Verzweigung sind, kann die rechtzeitige Datenversorgung von *B* vor Beginn seiner Ausführung nicht sichergestellt werden; d. h. *B* könnte bereits gestartet werden, bevor *C* das obligatorische Datenobjekt *d* geschrieben hat. Darüber hinaus wird auf *d* durch *B* lesend zugegriffen, während ein konkurrierender Schreibzugriff ausgehend von *C* oder *D* erfolgt. Auch dies wird aktuell von ADEPT2 verhindert, obwohl solche Konstellationen aus fachlicher Sicht durchaus Sinn machen könnten. Z. B. werden in Kliniken häufig Kurzarztbriefe verschickt, damit der niedergelassene Arzt, der für die Nachsorge zuständig ist, rechtzeitig über den Ausgang der stationären Therapie informiert ist. Während also eine Vorabversion des eigentlichen Arztbriefes bereits abgesendet wurde, kann nebenläufig die weitere Bearbeitung des Briefes stattfinden; dies umfasst auch oftmals einige Korrektur-

schritte. In ADEPT2 hingegen müssen der Zeitpunkt des Abschlusses des Kurzarztbriefes und der Übergang zur Erstellung der ausführlichen Version genau definiert sein. Dies kann z. B. erreicht werden, indem sich die Datenobjekte für Kurzarztbriefe und Arztbriefe voneinander unterscheiden oder indem die Reihenfolge, in der der lesende und schreibende Zugriff auf das Datenobjekt erfolgt, festgelegt wird.

Abbildung 95 zeigt, wie der zugehörige SPF-Typ-Graph gestaltet werden könnte, um durch Spezifikation einer eindeutigen Ausführungsreihenfolge das Zustandekommen der oben aufgezeigten Problematik zu verhindern.

Abbildung 95 Beispiel für die Transformation eines SPF-Graphen unter Beachtung von Datenflussabhängigkeiten



Im Hinblick auf den Datenflussaspekt ist das Prozessfragment *A* unabhängig von den anderen Prozessfragmenten. Aus diesem Grund werden im SPF-Typ-Graphen keine Constraint-Relationen zwischen dem Knoten für *A* und den anderen Knoten definiert. *B* hingegen benötigt ein Datenobjekt *d*, das entweder von *C* oder *D* geschrieben werden muss. Die Ausführung beider Prozessfragmente *C* und *D* ergibt keinen Sinn, da *B* entweder auf das Objekt zugreift, nachdem es von *C* geschrieben worden ist oder nachdem es von *D* überschrieben wurde. ADEPT2 erfordert eine klare Vorgabe der Reihenfolge, in der Lese- und Schreibzugriffe auf das Datenobjekt stattfinden; daher kann dem Vorgängerknoten von *C* und *D* ein XOR-Operator zugeordnet werden. Darüber hinaus wird zwischen *B* und dem Kontextknoten *Y* ein Sequenz-Constraint definiert.

Eine spezielle Problematik ergibt sich mit Blick auf Mehrfachreferenzen in SPF-Typ-Graphen. Dabei handelt es sich um Knoten oder Teilgraphen, die von mehreren übergeordneten Kontextknoten referenziert werden. Dies hat große Vorteile im Hinblick auf Identifikationsmöglichkeiten identischer Teilgraphen auf Basis des Graphmorphismus. Bei der Konfiguration des SPF-Typ-Graphen lassen sich Mehrfachreferenzen nutzen, um sie wie Klone gleichartig zu konfigurieren. Auf diese Weise ist es möglich, den Konfigurationsaufwand stark zu reduzieren (siehe Kapitel 4.4.4.2). Dasselbe gilt später auch für die Rekonfiguration. Mit Blick auf die Konsistenz des Datenflusses ergeben sich jedoch zusätzliche Risiken. Mehrfachreferenzen im

SPF-Typ-Graphen erscheinen im SPF-Graphen als identische Knoten mit unterschiedlichen Vorgängern, die sich nur durch ihre spezielle Knotenidentität im SPF-Graphen voneinander unterscheiden. Während Klone mit demselben Vorgänger wie Constraint-relevante Knoten behandelt werden, gilt diese Einschränkung für Klone auf Basis von Mehrfachreferenzen nicht (siehe Kapitel 4.5.3); das bedeutet, dass gleichartige Prozessfragmente auch parallel zueinander stattfinden können. Dies ist kritisch, da bei diesen Fragmenten eine erhöhte Wahrscheinlichkeit dafür besteht, dass sie auf dieselben Datenobjekte lesend und schreibend zugreifen. Damit ist die Gefahr groß, dass bei der Transformation die in ADEPT2 unerwünschten konkurrierenden Lese- und Schreibzugriffe durch zueinander parallel ausführbare Aktivitäten entstehen. Dieses Problem kann gelöst werden, indem für solche Arten von Prozessfragmenten unterschiedliche Datenobjekte erzeugt werden, auf denen sie unabhängig voneinander operieren können. Sei $SG = (V, E, V_T, V_{NT}, ...)$ mit $V_{Trans} \subseteq V$ der mit W assoziierte SPF-Graph. Beim Ergänzen des Datenflusses im Fall von Mehrfachreferenzen gilt also die folgende Regelung:

$$\forall param \in Parameters^m \; \exists d \in D: (Name^{param} \cup param) = d \wedge (\exists u \in V_{Trans}: u \neq v_{Trans} \wedge g_v(u) = g_v(v_{Trans})) \wedge (\exists x, y \in V: x \rightarrow^* u \wedge y \rightarrow^* v_{Trans} \wedge x \neq y \wedge g_v(x) \neq g_v(y)) \Rightarrow addDataElements(W, Name^{param} \cup param \cup count, DataType^{param}, NULL)$$

Mehrfachreferenzen lassen sich dadurch erkennen, dass es mindestens zwei Knoten im SPF-Graph gibt, die auf denselben Knoten im SPF-Typ-Graph abgebildet werden und bei denen sich der Pfad der Kontextknoten bis zur Wurzel in mindestens einem Fall voneinander unterscheidet. Die Bezeichnung der Datenobjekte für solche Prozessfragmente weicht von der Generierung der Bezeichnung anderer Fragmente ab, indem die Vereinigungsmenge aus Parametername und Identität noch um eine Variable »count« erweitert wird. Diese Variable muss jeweils eindeutig sein, so dass sichergestellt werden kann, dass sich die Datenobjekte der Mehrfachreferenzen voneinander unterscheiden. Nun ergibt sich die Frage, wie bei Prozessfragmenten, die lesend oder schreibend auf das Datenobjekt einer Mehrfachreferenz zugreifen, entschieden werden kann, welches Datenobjekt sie aktuell nutzen sollen. In diesem Konzept hängt die Auswahl von der Aktualität des Datenobjektes ab; d. h. der Zugriff erfolgt auf das Objekt, das gemäß Prozessdefinition zuletzt geschrieben wird. Ist die Wahl nichtdeterministisch, so wird ein Objekt zufällig ausgewählt. Eine bessere Lösung könnte z. B. darin bestehen, dass über die Aktualität eines Datenobjektes nicht statisch zur Modellierungszeit, sondern dynamisch zur Laufzeit entschieden wird, oder dass Anwender zur Laufzeit selbst bestimmen, welches der zur Verfügung stehenden Datenobjekte sie nutzen wollen. Eine Analyse, in welchen Fällen solche erweiterten Konzepte nützlich sind, und deren Umsetzung werden in dieser Arbeit jedoch nicht durchgeführt. Die in Frage kommenden Aktivitäten im WSM Net können folgendermaßen identifiziert werden:

$$lastActivities = \{n \in N | \; n \in pred^*(Name^p \cup VID(v_{Trans})) \wedge (n, Name^{param} \cup param \cup count^x, write) \in DataE \wedge (\nexists \; m \in succ^*(n): (m, Name^{param} \cup param \cup count^y, write) \in DataE)\}$$

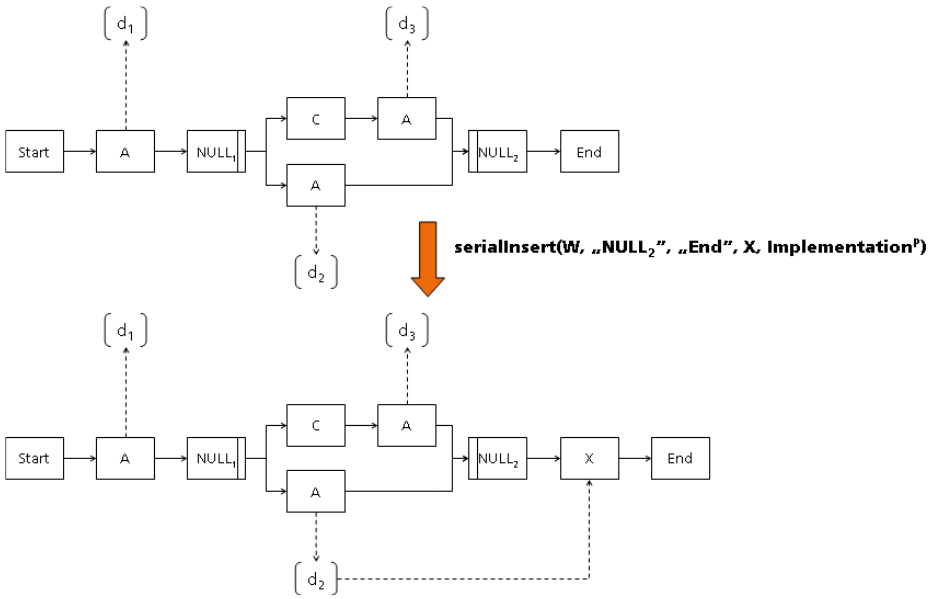
»$count^x$« bzw. »$count^y$« repräsentieren jeweils zwei verschiedene Werte aus der Wertemenge für die Variable »count«, die für dieses Datenobjekt bereits generiert wurden. Die Aktivitäten, die zuletzt schreibend auf das Objekt zugegriffen haben, werden dadurch ermittelt, dass sie keine Nachfolgeraktivitäten mehr haben, die dasselbe Kriterium erfüllen. Anschließend kann zwischen dem Datenobjekt, das von einer Aktivität aus der Menge *lastActivities* geschrieben wurde und der aktuell betrachteten Aktivität, für die die Datenflüsse ergänzt werden sollen, eine neue Datenkante eingezogen werden.

$\forall param \in Method^{p}: IOType^{param} \in \{IN, IN/OUT\} \wedge \exists d \in DW: d = Name^{param} \cup param \cup count^{*} \Rightarrow addDataEdges(W, (Name^{p} \cup VID(v_{Trans}), lastD, read))$ mit $lastD \in D, n \in lastActivities$ und $(n, lastD, write) \in DataE$

$\forall param \in Method^{p}: IOType^{param} \in \{OUT, IN/OUT\} \wedge \exists d \in DW: d = Name^{param} \cup param \cup count^{*} \Rightarrow addDataEdges(W, (Name^{p} \cup VID(v_{Trans}), lastD, write))$ mit $lastD \in D, n \in lastActivities$ und $(lastActivities, lastD, write) \in DataE$

Die folgende Abbildung demonstriert die Lösung an einem Beispiel.

Abbildung 96 Lösungsansatz zum Umgang mit Mehrfachreferenzen unter Vermeidung von Datenflussinkonsistenzen



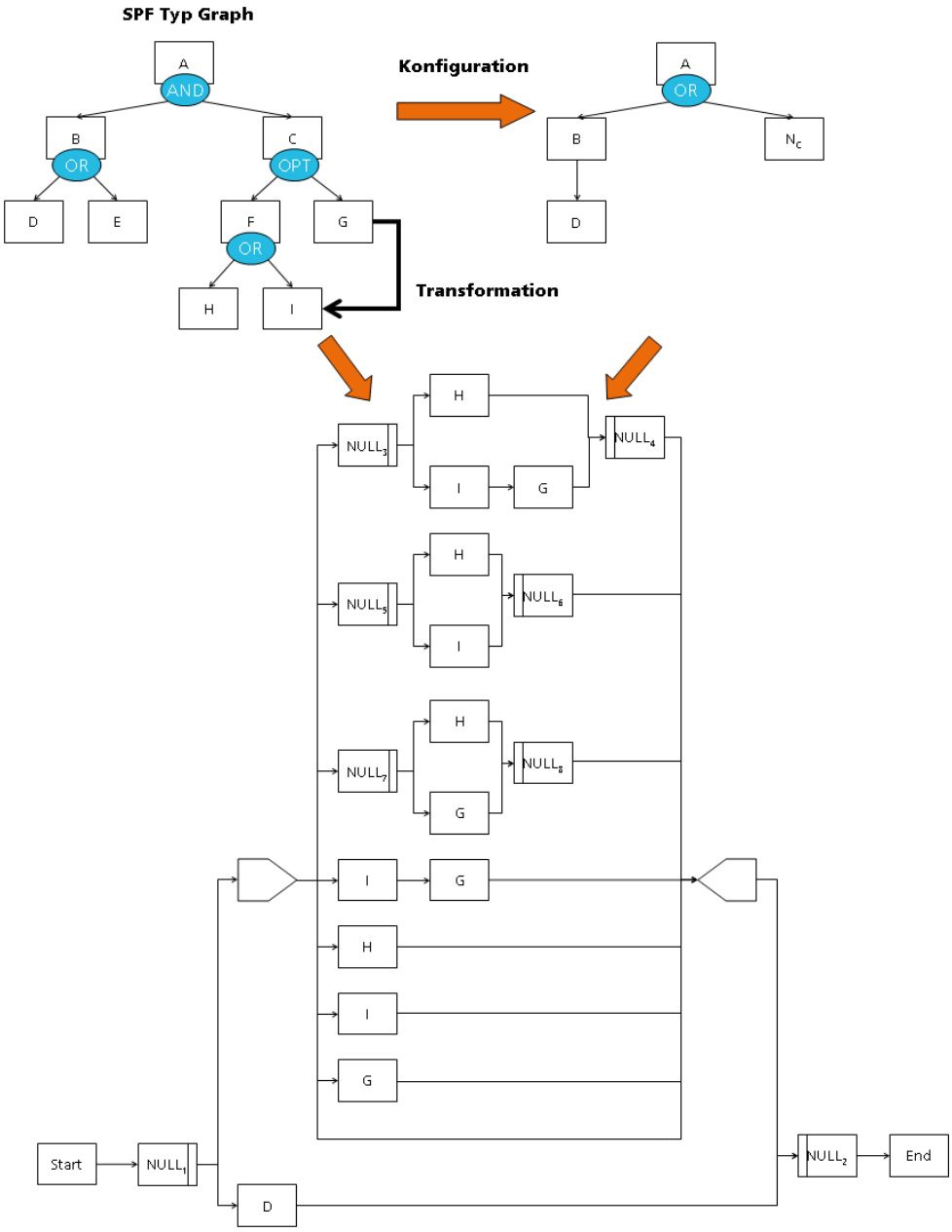
In dem Beispiel existieren insgesamt drei Klone von Prozessfragment *A*, die auf Mehrfachreferenzen im SPF-Typ-Graphen zurückgehen. Alle drei produzieren dasselbe Datenobjekt, für das bei der Transformation in ein WSM Net jedoch drei unterschiedliche Objekte generiert werden. Das neu einzufügende Prozessfragment *X* benötigt nun ein Objekt dieses Typs. Hierbei kommen die beiden Aktivitäten *A* in der parallelen Verzweigung in Frage, da sie jeweils das aktuellste Datenobjekt erzeugen. Bei der automatischen Ergänzung des Datenflusses, wird in diesem Lösungsansatz eines der zur Auswahl stehenden Datenobjekte zufällig selektiert und eine neue lesende Datenflusskante erzeugt. Speziell für diesen Zweck entwickelte Entscheidungsunterstützungssysteme könnten dazu beitragen, Automatismen bereitzustellen, um hier in Abhängigkeit bestimmter Kontextparameter eine adäquate Entscheidung herbeizuführen. Bei Bedarf müssen die Prozessfragmente, die auf die Datenobjekte der Mehrfachreferenzen lesend zugreifen auch so flexibel implementiert werden, dass sie eine beliebige Menge solcher Datenobjekte verarbeiten können. Eine weitere, einfach zu realisierende Alternative besteht darin, Mehrfachreferenzen bei der Transformation genau wie normale Klonknoten zu behandeln.

4.5.5 Transformation unvollständiger Konfigurationen von SPF-Typ-Graphen

Die Spezifikation der Regeln für die Transformation von SPF-Graphen in WSM Nets bezog sich bisher ausschließlich auf vollständige Konfigurationen; d. h. es gibt im SPF-Graphen nur noch terminale Knoten, wobei an die Blattknoten der entsprechenden Pendants im SPF-Typ-Graphen die in das WSM Net einzufügenden Prozessfragmente geknüpft sind. Wie sich gezeigt hat, repräsentieren WSM Nets auf Basis vollständig konfigurierter SPF-Graphen Prozessdefinitionen, die nur noch über sequentielle und parallele Kontrollflusskonstrukte sowie bei Bedarf über Synchronisationskanten verfügen. Da Schleifen über die Anzahl der Klone determiniert werden und bedingte Verzweigungen nicht vorgesehen sind, gibt es auf dieser Grundlage also keine Möglichkeit, zur Laufzeit Einfluss auf die Prozessausführung zu nehmen. Dies kann sich ändern, indem auch unvollständig konfigurierte SPF-Typ-Graphen in Prozessdefinitionen transformiert werden. Jeder nicht-terminale Knoten im SPF-Graphen entspricht dann einer variablen Einheit, deren konkrete Ausgestaltung erst zur Laufzeit festgelegt wird. So lässt sich z. B. definieren, dass im Rahmen der Diagnostik eine radiologische Untersuchung durchgeführt werden soll; welche Untersuchungen für den jeweiligen Patienten konkret erfolgen, wird erst zur Laufzeit vom behandelnden Arzt entschieden. Die Auswahl kann dann natürlich nur innerhalb der Grenzen des Möglichen, also der zur Verfügung stehenden Prozessfragmente, vorgenommen werden. Dieses Vorgehen bietet sich an, wenn Teile des klinischen Pfades einfach zu variabel sind, um einen eindeutigen Standard vorzugeben und die konkrete Entscheidung ausschließlich im Ermessen des Arztes liegt und in Abhängigkeit des jeweiligen Behandlungsfalles jedes Mal neu getroffen werden muss.

Die meisten Prozessmodellierungssprachen, so auch ADEPT2, unterstützen bedingte Verzweigungskonstrukte; in diesem Fall wird zur Laufzeit ausgewertet, welcher Ausführungszweig für einen konkreten Behandlungsfall gewählt werden soll. Der Aufbau bedingter Verzweigungen im Rahmen der Transformation eines SPF-Graphen in ein WSM Net macht es erforderlich, dass alle Informationen zu den möglichen Alternativen verfügbar sind. SPF-Graphen stellen diese Informationen jedoch nicht bereit, da die Existenz eines nicht-terminalen Knotens lediglich signalisiert, dass zur Modellierungszeit das Wissen, das zur Konfiguration des Knotens notwendig ist, noch nicht vorhanden ist. Informationen zu alternativen Ausführungswegen sind jedoch Bestandteil des SPF-Typ-Graphen und der Graphgrammatik. Die Generierung bedingter Verzweigungsstrukturen erfordert demnach nicht nur die Transformation der terminalen Bereiche von SPF-Graphen, sondern auch die Transformation von Teilen des SPF-Typ-Graphen, die im SPF-Graphen noch unspezifiziert sind. Dieses Vorgehen hat prinzipiell den Vorteil, dass das gesamte Prozessmodell und seine Varianten erzeugt werden können, auch wenn das WfMS keine Möglichkeiten zur Ad-hoc-Änderung von Prozessinstanzen bietet und damit auch die Rekonfigurationsphase des Lösungskonzeptes nicht unterstützt. Es erhöht jedoch die Komplexität des Transformationsvorgangs beträchtlich, da die Informationen zur Generierung prozeduraler Modellstrukturen nun aus zwei deklarativen Modellen, nämlich dem SPF-Typ-Graphen und dem SPF-Graphen, stammen müssen; bisher wurde der SPF-Typ-Graph während der Transformationsphase lediglich benötigt, um Abhängigkeiten zwischen Knoten festzustellen und die in die Prozessdefinition einzufügenden Prozessfragmente zu identifizieren. Darüber hinaus wirkt sich dieses Vorgehen aber auch auf die Komplexität der resultierenden Prozessdefinition aus. Die nachfolgende Abbildung illustriert beispielhaft die Transformation nicht-terminaler Knoten in bedingte Verzweigungsstrukturen.

Abbildung 97 Transformation eines nicht-terminalen Knotens in eine prozedurale, bedingte Verzweigungsstruktur



Die oben dargestellte Prozessdefinition als WSM Net enthält noch nicht die Datenobjekte und Datenflüsse, die notwendig wären, um die Entscheidung über den zu durchlaufenden, alternativen Zweig zu treffen. Zu diesem Zweck wären spezielle Prozessfragmente notwendig, die nur dazu dienen, die für die Entscheidung notwendigen Datenobjekte zu produzieren. Das WfMS müsste dann die Interpretation der vom Prozessfragment als Ergebnis zurückgelieferten Datenobjekte übernehmen; das bedeutet aber, dass Datenobjekte nicht mehr generell als »Black Box« behandelt werden können, was sich äußerst nachteilig auf die Unabhängigkeit des hier vorgestellten Lösungskonzeptes auf konkrete WfMS auswirkt. Auffallend ist die Komplexität, die bereits aus der Umwandlung eines flachen Teilbaums resultiert. Sie wird noch zusätzlich erhöht, sobald Constraint-Relationen zu im SPF-Graphen enthalten-

den terminalen Knoten oder zu bisher nur im SPF-Typ-Graphen vorhandenen Knoten definiert sind. Eine große, zusätzliche Schwachstelle dieser Lösungsalternative besteht darin, dass Teile der Prozessdefinition nicht mehr mit den Knoten des SPF-Graphen korrespondieren. Sobald sich der Fachanwender für einen Ausführungszweig entschieden hat, müsste der SPF-Graph nachträglich konfiguriert werden; soll sich die Bezeichnung der Prozessaktivitäten weiterhin aus dem Namen des Prozessfragments und der Identität des Knoten im SPF-Graphen zusammensetzen, müsste sie nachträglich angepasst werden. Die Rekonfiguration von bedingten Verzweigungsstrukturen ist grundsätzlich nicht möglich, da die entsprechenden Knoten nicht im SPF-Graphen existieren; dies trägt nicht zur Transparenz des Rekonfigurationsprozesses für die Fachanwender bei. Aus diesen Gründen repräsentiert die Generierung bedingter Verzweigungsstrukturen auf Basis von SPF-Typ-Graphen also nicht das Mittel der Wahl.

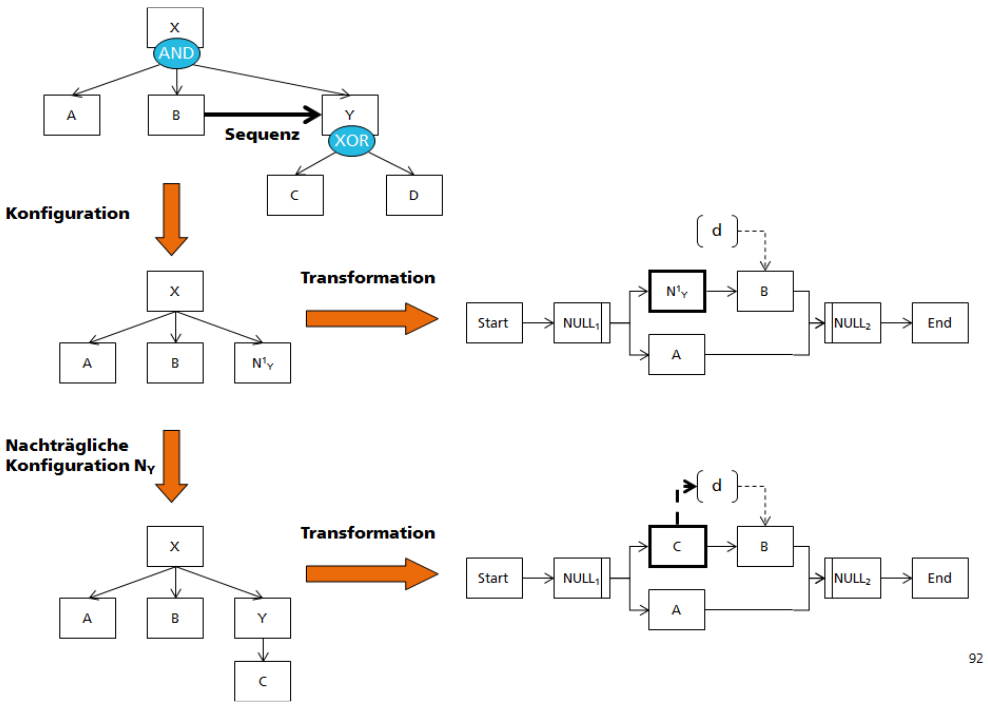
Eine Alternative zur Erzeugung hoch-komplexer bedingter Verzweigungsstrukturen aus deklarativen SPF-Typ-Graphen besteht in der direkten Einbettung der nicht-terminalen Knoten in der Prozessdefinition und ihrer nachträglichen Konfiguration. Um dies zu erreichen, werden die nicht-terminalen Knoten im SPF-Graphen wie terminale Blattknoten behandelt. Das bedeutet, auch für sie wird ein Element in die Prozessdefinition eingefügt. Dabei kann es sich allerdings nicht um ein normales, ausführbares Prozessfragment handeln, da Nichtterminalknoten keine direkte Repräsentation im SPF-Typ-Graphen besitzen. Auch ist bei dem entsprechenden terminalen Knoten nicht sichergestellt, dass es sich dabei um einen SPF-Knoten handelt, der mit einem Prozessfragment assoziiert ist. Als mögliche Realisierungsvarianten kommen jedoch all diejenigen Konstrukte in Frage, die Change Pattern für Änderungen in vordefinierten Regionen umsetzen [Weber et al. 2008]. Diese zweite Form von Change Pattern bewirkt keine strukturellen Anpassungen an Prozessdefinitionen bzw. deren Instanzen, wie die in Kapitel 4.5.2 beschriebenen ADEPT2 Änderungsoperationen; stattdessen lassen sie die konkrete Ausgestaltung definierter Regionen in der Prozessdefinition offen. Die Modellierung des Prozesses wird also erst zur Laufzeit finalisiert. Dieser Ansatz hat den Vorteil, dass die Variabilität im Prozess auf exakt festgelegte Regionen reduziert ist und die Planbarkeit des Behandlungsablaufs nur in Abhängigkeit der unspezifizierten Region eingeschränkt wird. Darüber hinaus ist diese Region bereits zur Instanziierungszeit der Prozessdefinition bekannt, so dass Zeitpunkt und Umfang der Änderung im Voraus kalkuliert werden können. Ad hoc Änderungen an der Prozessstruktur sind hingegen nicht planbar; die Regionen, die von solchen Änderungen betroffen sind, sind im Voraus unbekannt.

Weber et al. haben insgesamt vier verschiedene solcher Change Pattern für Änderungen in vordefinierten Regionen identifiziert [Weber et al. 2008]:
- Nachträgliche Selektion von Prozessfragmenten: Die konkrete Implementierung eines Prozessfragments, wie z. B. ein Subprozess oder ein Web Service, wird erst zur Laufzeit festgelegt (analog zu Late Binding, siehe Kapitel 3.2.2).
- Nachträgliche Modellierung von Prozessfragmenten: Bei diesem Ansatz werden die fehlenden Prozessfragmente zur Laufzeit ergänzt, wobei dem Verantwortlichen dieselben Freiheiten und Werkzeuge zur Verfügung stehen wie bei der normalen Prozessmodellierung (analog zu Late Modeling, siehe Kapitel 3.2.3).
- Nachträgliche Komposition von Prozessfragmenten: In Abhängigkeit bereits spezifizierter Rahmenbedingungen wird aus einer Menge vordefinierter Prozessfragmente ausgewählt; darüber hinaus werden die selektierten Fragmente in eine Ausführungsreihenfolge gebracht (analog zu Late Composition, siehe Kapitel 3.2.3).
- Mehrfache Instanziierung eines Prozessfragments: Es wird zur Laufzeit entschieden, wie viele Instanzen von einem Fragment gebildet werden sollen.

In Kapitel 3.2.3 wurden Pockets of Flexibility (PoF) als ein Ansatz für Late Modeling vorgestellt. Pockets of Flexibility werden wie normale Aktivitäten in die Prozessdefinition eingefügt. Es handelt sich hierbei jedoch um Platzhalter, deren Inhalt zur Laufzeit modelliert werden muss; d. h. sobald die Ausführung an einer Platzhalteraktivität angekommen ist, muss die Modellierung dynamisch durchgeführt und abgeschlossen werden, bevor die weitere Abarbeitung des Prozesses vorgenommen werden kann. Gemäß diesem Lösungsansatz könnte für jeden Nichtterminalknoten in der Prozessdefinition eine Platzhalteraktivität eingefügt werden. Die Umsetzung des Change Pattern würde dann der Konfiguration des nicht-terminalen Knotens mit anschließender Transformation des SPF-Graphen entsprechen. Mit der Generierung einer Graphgrammatik und der Bereitstellung von Algorithmen zur Sicherstellung von Constraints wird somit ein Beitrag zur Formalisierung der Change Pattern zur Durchführung von Änderungen in vordefinierten Regionen geliefert. Im Unterschied zur ursprünglichen Definition der Change Pattern müssen sich die Änderungen durch die nachträgliche Konfiguration allerdings nicht nur auf die durch die Platzhalteraktivität eingeschränkte Region beziehen. Je nachdem welche Constraint-Relationen definiert sind, können die Stellen, an denen die neuen Prozessfragmente eingefügt werden, auch variieren. Soll die Region aber statisch sein, so dürfen die Nachfolger des entsprechenden Kontextknotens im SPF-Typ-Graphen nicht über Anforderungs- oder Sequenz-Constraints mit Knoten anderer Teilbäume verknüpft sein. ADEPT2 und WSM Nets verfügen nicht über Möglichkeiten zur Unterstützung der hier genannten Change Pattern durch die Umsetzung von Konzepten wie PoF oder Worklets. Dennoch sollen die Realisierungsoptionen in dieser Arbeit zumindest konzeptionell beschrieben werden.

Das Change Pattern zur nachträglichen Selektion von Prozessfragmenten lässt sich realisieren, indem der Nichtterminalknoten über Regeln der Graphgrammatik in einen Kontextknoten umgewandelt werden kann, der über einen XOR-, OR- oder OPT-Operator verfügt und dessen Nachfolger auf SPF-Knoten im SPF Typ Knoten abgebildet werden (siehe Abbildung 98).

Abbildung 98 Umsetzung des Change Pattern zur nachträglichen Selektion von Prozessfragmenten



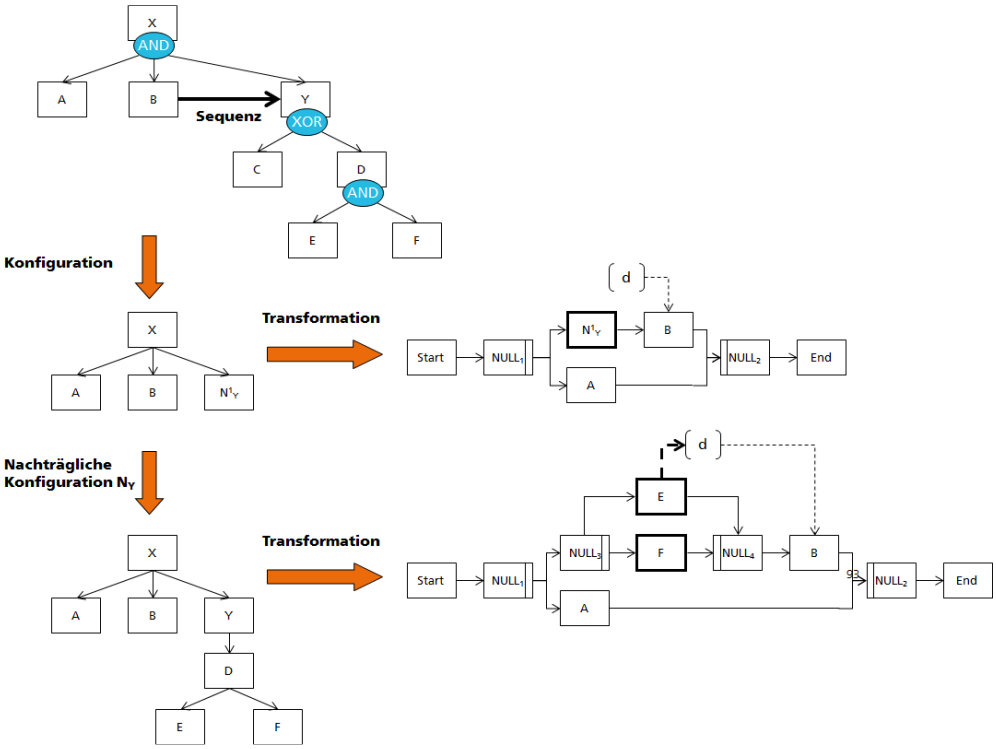
92

In dem obigen Beispiel wurde ein einfacher SPF-Graph in eine Prozessdefinition transformiert, wobei N^1_Y einem nicht-terminalen Knoten entspricht. Für diesen Knoten wurde im Rahmen der Transformation eine Platzhalteraktivität generiert, die wie ein normales Prozessfragment nach den Transformationsregeln in die Prozessdefinition eingefügt wurde. Der Nichtterminalknoten kann nun entweder noch vor der Instanziierung der Prozessdefinition oder zur Laufzeit beim Start der Platzhalteraktivität nachträglich konfiguriert werden. Gemäß der Regel der Graphgrammatik wurde N^1_Y durch den terminalen Knoten *Y* und seinen Nachfolger N^1_C ersetzt; da N^1_C nur noch auf einen terminalen Knoten abgebildet werden kann, erfolgt die Regelanwendung automatisch. Dementsprechend wird also Prozessfragment *C* vor *B* in die Prozessdefinition eingefügt. Die manuelle Auswahl des Prozessfragments könnte bei Bedarf auch wie bei dem Worklet Ansatz in Kapitel 3.2.2 durch Ripple-Down-Rules unterstützt werden. An diesem Beispiel zeigt sich auch, dass die nachträgliche Konfiguration Auswirkung auf die Konsistenz des Datenflusses haben kann. So erfordert Aktivität *B* ein Datenobjekt, das von einem der Prozessfragmente *C* oder *D* erzeugt werden muss. Solange die Platzhalteraktivität in der Prozessdefinition eingebettet ist, kann das Objekt jedoch nicht geschrieben werden; das bedeutet, dass die Prozessdefinition erst nach der Konfiguration von N^1_Y den Korrektheitskriterien von WSM Nets in ADEPT2 entspricht.

Der Late Modeling Ansatz kann nicht so einfach unterstützt werden, da Fachanwender in der Lage sein sollen, selbst neue Prozessfragmente zu definieren, um sie anschließend in die Prozessdefinition einzubetten. In diesem Fall müssten also zunächst die neuen Fragmente implementiert und über eine Erweiterung der Knoten- und Kantenmenge in den SPF-Typ-Graphen integriert werden. Mit der Aktualisierung des SPF-Typ-Graphen, verändert sich jedoch auch die Graphgrammatik. Der Late Modeling Ansatz würde daher eine Übertragung des SPF-Graphen auf Basis der alten Grammatik auf die neue Grammatik bedeuten. Dies entspricht der Durchführung einer Prozessevolution. Rinderle hat am Beispiel von WSM Nets einen Formalismus zur Durchführung der Prozessevolution entwickelt [Rinderle 2004]; es wäre interessant zu untersuchen, inwieweit sich die von ihr vorgestellten Methoden auf SPF-Typ-Graphen und deren Konfigurationen übertragen lassen (siehe Kapitel 8.4). Da zur Laufzeit allerdings ärztliches, pflegerisches oder administratives Personal für die Prozessanpassungen verantwortlich ist und von diesem nicht erwartet werden kann, eigenständig Prozessfragmente zu implementieren, wird die Umsetzung des Late Modeling an dieser Stelle nicht vertieft.

Im Gegensatz zu Late Modeling kommt das Change Pattern zur nachträglichen Komposition von Prozessfragmenten dem hier vorgestellten Lösungsansatz sehr entgegen. Die Komposition entsteht durch die nachträgliche Konfiguration des nicht-terminalen Knoten im SPF-Graphen. Anschließend wird der Graph wieder in die Prozessdefinition überführt. Abbildung 99 illustriert den Vorgang an einem Beispiel.

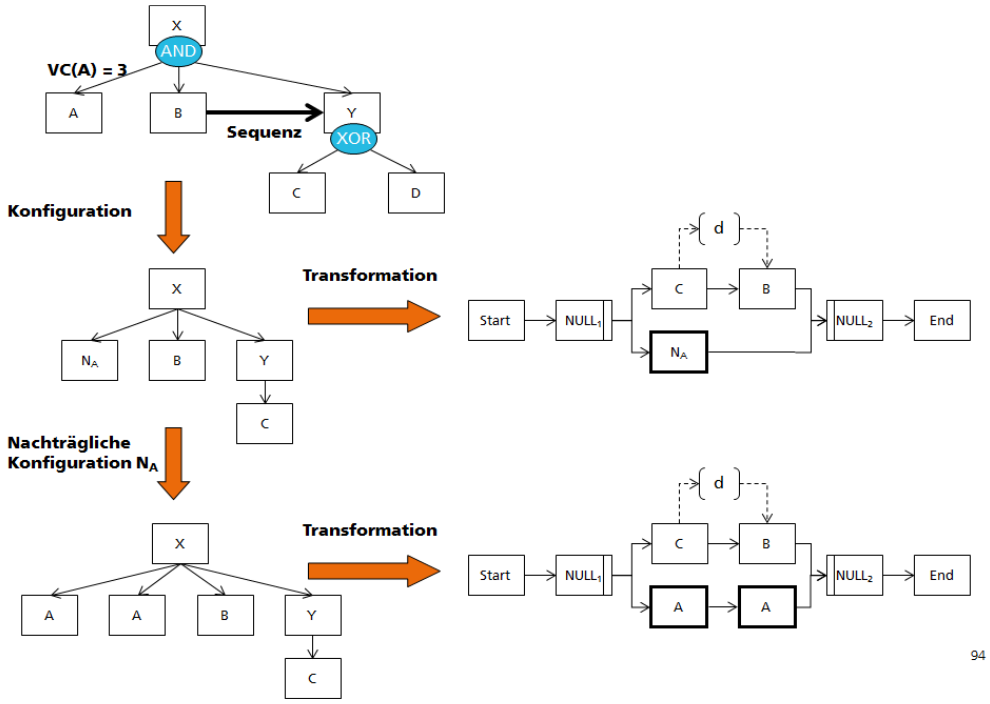
Abbildung 99 Umsetzung des Change Pattern zur nachträglichen Komposition von Prozessfragmenten



Das Change Pattern zur nachträglichen Komposition von Prozessfragmenten lässt sich auf dieselbe Art und Weise wie das Pattern zur nachträglichen Selektion umsetzen. Der wesentliche Unterschied besteht darin, dass es sich bei den Nachfolgern des Nichtterminalknotens im SPF-Typ-Graphen auch um Kontextknoten handeln kann. In dem Beispiel wird zunächst Knoten *D* ausgewählt; da diesem ein AND-Operator zugeordnet ist und es für *E* und *F* keine Klone geben kann, werden die nachfolgenden Konfigurationsschritte automatisch ausgeführt. Aufgrund der Tatsache, dass sonst keine Abhängigkeiten definiert sind, werden die Prozessfragmente von *E* und *F* parallel zueinander in das WSM Net eingefügt.

Auch das letzte Pattern für die Bildung multipler Instanzen lässt sich einfach realisieren. In diesem Fall entspricht der zu konfigurierende Knoten einem SPF-Knoten, für den Klone gebildet werden können. Zu einem späteren Zeitpunkt kann nun entschieden werden, wie viele Klone erstellt werden sollen. Konform zu den Transformationsregeln werden alle Klone jedoch sequentiell hintereinander in die Prozessdefinition eingefügt und können auch nur der Reihe nach ausgeführt werden. Soll die Möglichkeit bestehen, die Klone nebenläufig auszuführen, müsste der SPF Knoten von mehreren übergeordneten Kontextknoten referenziert werden (Mehrfachreferenz), die wiederum Nachfolger eines gemeinsamen Kontextknoten sind; diesem gemeinsamen Kontextknoten würde dann ein OR oder ein OPT-Operator zugewiesen, so dass die Menge der Kindknoten dynamisch festgelegt werden kann. Wie bereits in Kapitel 4.5.4 dargestellt, können Mehrfachreferenzen jedoch zu Problemen mit der Integrität des Datenflusses führen. Die nächste Abbildung verdeutlicht die Art zur Umsetzung im Fall eines Klonknoten beispielhaft.

Abbildung 100 Umsetzung des Change Pattern zur mehrfachen Instanziierung eines Prozessfragments



Wie das Beispiel zeigt, wurde für den nicht-terminalen Knoten N_A eine Platzhalteraktivität in der Prozessdefinition angelegt. In einem späteren Schritt kann nun N_A über Anwendung der entsprechenden Regeln der Graphgrammatik in bis zu drei Klone umgewandelt werden. In dem Beispiel wurden zwei terminale Knoten von *A* erstellt, so dass im WSM Net an der Stelle der Platzhalteraktivität nun die zwei Klon-Aktivitäten in sequentieller Reihenfolge erscheinen.

In den Beispielen zur nachträglichen Konfiguration von nicht-terminalen Knoten wird bisher immer davon ausgegangen, dass ein SPF-Graph das erste Mal in eine Prozessdefinition überführt wird (»build-time«). Die hier genannten Change Pattern kommen allerdings zur Laufzeit, also nach der Instanziierung des Prozesses zum Einsatz (»runtime«). Das bedeutet, dass bei der Transformation des SPF-Graphen im Anschluss an die nachträgliche Konfiguration des Nichtterminalknotens der Ausführungsstatus der Prozessinstanz berücksichtigt werden muss. Wie nach der Rekonfigurationsphase müssen also bei der erneuten Transformation Änderungen an der Prozesshistorie verhindert werden. Grundsätzlich kann dies vermieden werden, wenn die Nachfolgerknoten des nachträglich konfigurierten Nichtterminalknotens über keine Constraint-Relationen in andere Teilbäume verfügen. Andernfalls müssen bei der nachträglichen Konfiguration mit anschließender Transformation des SPF-Graphen dieselben Rahmenbedingungen berücksichtigt werden wie bei der Rekonfiguration, die in Kapitel 4.6 behandelt wird. Ansonsten ist die Berücksichtigung von Constraint-Relationen bei der Nutzung von Change Pattern in vordefinierten Regionen deutlich einfacher möglich als bei der Erzeugung konditionaler Verzweigungen. Angenommen, der SPF-Graph enthält einen Knoten, der in Konflikt steht mit einem Nachfolger des nicht-terminalen Knotens; die Konfiguration des nicht-terminalen Knotens basiert dann auf einer temporären Grammatik, die verhindert, dass der Konflikt auslösende Knoten nachträglich in die Prozessdefinition integriert wird.

4.5.6 Zusammenfassung der Ergebnisse

In diesem Kapitel wurde gezeigt, wie ein SPF-Graph in eine konkrete Prozessdefinition überführt werden kann. Da es unterschiedliche Modellierungssprachen und WfMS gibt, die als Zielsprache bzw. Zielsystem genutzt werden können, ist der Transformationsschritt selbst nicht generisch, sondern spezifisch für die aktuell gewählte Ausführungsumgebung. Aus diesem Grund wurden mit WSM Nets und ADEPT2 eine konkrete Prozessmodellierungssprache und ein konkretes WfMS selektiert, an deren Beispiel der Transformationsprozess verdeutlicht worden ist. WSM Nets entsprechen einer mengenbasierten Spezifikation für Prozessdefinitionen, die von ADEPT2 ausgeführt werden können. Neben dieser Spezifikation und einer Reihe von Korrektheitskriterien für WSM Nets stellt ADEPT2 eine Menge von Änderungsoperationen auf unterschiedlichen Granularitätsebenen bereit, deren konsequente Anwendung zu strukturell korrekten WSM Nets führt. Es wurde untersucht, welche Änderungsoperationen genutzt werden können, um die Transformationsphase zu unterstützen. Dabei wurden existierende Änderungsoperationen identifiziert und zwei hochwertige Änderungsoperationen zusätzlich definiert.

Nach der Klärung der konzeptionellen Grundlagen von WSM Nets und der Auswahl von Änderungsoperationen wurden die Regeln zur Umwandlung deklarativer SPF-Graphen in prozedurale WSM Nets formal beschrieben. Währenddessen wurden auch Restriktionen festgestellt, die es schwierig machen, deklarative Prozessabläufe in einer imperativen Logik auszudrücken. Eine Einschränkung ergibt sich durch die strenge Strukturierung von WSM Nets in Form von Kontrollblöcken; das ist notwendig, um die strukturelle Integrität der Prozessdefinition effizient überprüfen zu können. Es erschwert aber die Transformation von Prozessfragmenten, für die im WSM Net bereits Aktivitäten existieren, die sowohl unabhängig als auch abhängig von dem neuen Fragment sind; dies gilt insbesondere dann, wenn sich die minimalen Kontrollblöcke der abhängigen und unabhängigen Aktivitäten überschneiden. In ADEPT2 wird mit Synchronisationskanten jedoch ein Konstrukt zur Verfügung gestellt, das es ermöglicht, die strenge Kontrollblockstrukturierung zu durchbrechen. Im Rahmen dieser Arbeit hat sich gezeigt, welch große Bedeutung Synchronisationskanten bei der automatischen Generierung von prozeduralen und korrekten Prozessdefinitionen auf Basis deklarativer Modelle haben können. Prinzipiell ließen sich alle Transformationsschritte auf paralleles Einfügen des Prozessfragments und Ergänzung notwendiger Synchronisationskanten und damit auf die drei hochwertigen Änderungsoperationen »insertBeforeNodeSet«, »insertBetweenNodeSet« und »insertAfterNodeSet« reduzieren. Da dies jedoch die Generierung extrem unübersichtlicher Prozessdefinitionen in ADEPT2 zur Folge hätte, wurden Regeln eingeführt, die eine größere Sequentialisierung der resultierenden WSM Nets bewirken sollen.

Schließend wurde ein einfaches Verfahren vorgestellt, wie die Prozessdefinition um Datenobjekte und Datenflüsse ergänzt werden kann, die durch die Schnittstellenspezifikationen der eingefügten Prozessfragmente bekannt sind. Dabei wurden auch mögliche Risiken im Hinblick auf die Integrität des Datenflusses aufgezeigt. Probleme ergeben sich z. B. dann, wenn nicht sichergestellt werden kann, dass alle Aktivitäten mit denen von ihnen benötigten Datenobjekten versorgt werden. Darüber hinaus müssen nichtdeterministische und damit möglicherweise konkurrierende Schreibzugriffe auf dieselben Datenobjekte in ADEPT2 verhindert werden. Beide Problemstellungen können durch adäquate Strukturierung von SPF-Typ-Graphen und die Spezifikation von Constraint-Relationen adressiert werden. Schließlich ergeben sich noch besondere Risiken durch die Zulassung von Mehrfachreferenzen im SPF-Typ-Graphen; dementsprechend können typgleiche Aktivitäten im WSM Net auch parallel zueinander ausgeführt werden. Dabei erhöht sich die Wahrscheinlichkeit für parallele Schreib- und Lesezugriffe und

somit das Risiko für inkonsistente Datenflüsse. Eine Möglichkeit besteht darin, für Prozessfragmente, die auf Mehrfachreferenzen zurückgehen, unterschiedliche Datenobjekte zu generieren. Andere Fragmente, die auf diese Objekte zugreifen, können dann standardmäßig immer das zuletzt aktualisierte Datenobjekt verwenden. Ist diese Auswahl nichtdeterministisch, so müssen gegebenenfalls verfeinerte Techniken entwickelt werden, die die Fachanwender bei der Auswahl eines Datenobjekts unterstützen.

Zum Schluss wurde gezeigt, wie die Transformation unvollständig konfigurierter SPF-Typ-Graphen dazu beitragen kann, Change Pattern zur Durchführung von Änderungen in vordefinierten Regionen umzusetzen. Auf diese Weise ist es möglich, Entscheidungen im Hinblick auf die konkrete Prozessgestaltung erst zur Laufzeit zu treffen. Sowohl die nachträgliche Selektion von Prozessfragmente, als auch die nachträgliche Komposition der Fragmente und die flexible Festlegung der Anzahl an Instanzen lassen sich durch Konfiguration des entsprechenden Nichtterminalknotens und erneute Transformation des SPF-Graphen realisieren. Zusammenfassend konnten die folgenden Ergebnisse produziert werden:

- Identifikation von Änderungsoperationen zur Realisierung der Transformationsphase unter Nutzung von ADEPT2
- Vorschlag für die Ergänzung von ADEPT2 um zwei zusätzliche, hochwertige Änderungsoperationen und Spezifikation der entsprechenden Algorithmen
- Entwurf eines formalen Regelwerks zur Umwandlung deklarativer Modelle in prozedurale Prozessdefinitionen mit dem Ziel eines möglichst hohen sequentiellen Anteils
- Beschreibung einer Methode zur automatischen Ergänzung von Datenobjekten und Datenflüssen auf Basis der Schnittstellenspezifikation von Prozessfragmenten
- Erarbeitung von Lösungen zur Sicherstellung der Integrität des Datenflusses durch Vermeidung unversorgter Aktivitäten, konkurrierender Schreibzugriffe auf Datenobjekte und Bereitstellung zusätzlicher Datenobjekte im Fall von Mehrfachreferenzen
- Konzeption von Methoden zur Unterstützung der Change Pattern für Änderungen in vordefinierten Regionen durch Transformation unvollständiger Konfigurationen von SPF-Typ-Graphen

4.6 Rekonfigurationsphase

Das im Rahmen dieser Arbeit entwickelte Lösungskonzept, bestehend aus den Phasen Definition, Konfiguration und Transformation, macht es möglich, Prozesswissen aus einer Domäne in einem Modell zu verstetigen, daraus für eine Vielzahl von Anwendungsfällen konkrete Prozesse abzuleiten und diese unter Nutzung einer gewünschten Modellierungssprache in ausführbare Prozessdefinitionen zu überführen. Durch die Entwicklung von domänenspezifischen Prozessmodellierungssprachen und Werkzeugen wie dem SPOT-Care-Plan-Modeler (siehe Kapitel 4.1.1) werden Fachexperten in die Lage versetzt, Geschäftsprozesse aus ihrer Sicht zu modellieren und auf Knopfdruck zur Ausführung zu bringen. Das Ziel dieser Arbeit besteht jedoch auch darin, es Anwendern nach der Instanziierung einer Prozessdefinition zu ermöglichen, die Instanz flexibel und mit geringem Aufwand zu verändern, ohne die Prozessausführung längere Zeit zu unterbrechen, um auf das Know-How von IT-Experten zurückzugreifen. Dies ist für die adäquate Unterstützung klinischer Pfade unbedingt notwendig, da Behandlungsabläufe oftmals nicht statisch strukturiert sind, sondern variabel in Abhängigkeit der individuellen Situation des Patienten, der ärztlichen Erfahrung sowie der internen und externen organisatorischen Rahmenbedingungen gestaltet werden müssen (siehe Kapitel 2.2). Obwohl klinische Pfade Standards für die stationäre Patientenversorgung ausgehend von einem Symptombild, einer Diagnose oder einer Therapie vorgeben, müssen Abweichungen von dem

geplanten Verlauf jederzeit durchführbar sein und durch das zugrunde liegende WfMS bestmöglich unterstützt werden.

Die Rekonfigurationsphase schafft den Rahmen, um Prozessdefinitionen und ihre Instanzen dynamisch an geänderte Situationen anzupassen. Sie dient der Realisierung manueller ad hoc Änderungen an Prozessen; dabei muss natürlich beachtet werden, dass die Änderungen nur innerhalb der Grenzen des durch den SPF-Typ-Graphen aufgespannten Möglichkeitsraums stattfinden können. Die Rekonfiguration bezieht sich nicht direkt auf die Prozessdefinition, sondern auf den SPF-Graphen, auf dem der Prozess basiert; auf diese Weise ist das Prinzip der Rekonfiguration unabhängig von konkreten Prozessmodellierungssprachen und WfMS anwendbar. Es werden Mittel bereitgestellt, die es ermöglichen, einzelne Knoten oder eine Menge gleichartiger Knoten nach bestimmten Kriterien auszuwählen und die Struktur des mit ihnen assoziierten Teilbaums zu modifizieren. Dabei kann es vorkommen, dass nicht die gesamte Struktur geändert werden soll, sondern sich die Adaption nur auf einen kleinen Teil des Prozesses bezieht; dies ist der Fall, wenn z. B. von einem Knoten zusätzliche Klone erzeugt, einzelne Nachfolger hinzugefügt oder entfernt werden sollen. Damit in solchen Fällen die gesamte Struktur des zu erhaltenden Teilgraphen nicht noch ein Mal festgelegt werden muss, wird auch ein strukturerhaltender Mechanismus zur Rekonfiguration von Knoten des SPF-Graphen vorgestellt. Jede Rekonfiguration hat die erneute Transformation des SPF-Graphen zur Folge; um den Transformationsaufwand in Grenzen zu halten, soll allerdings nicht der gesamte Graph umgewandelt werden, sondern lediglich der Teil, auf den sich die Änderung erstreckt.

Die Rekonfiguration bezieht sich ausschließlich auf SPF-Graphen als Ableitungen von SPF-Typ-Graphen und nicht auf konkrete Prozessdefinitionen wie WSM Nets. Die in diesem Kapitel vorgestellten Konzepte zur Rekonfiguration sind daher prinzipiell unabhängig von WfMS-spezifischen Lösungen und Werkzeugen einsetzbar. Die Berücksichtigung spezieller Systeme ist erst bei der dynamischen Adaption von Prozessinstanzen zur Laufzeit und bei der erneuten Transformation des veränderten SPF Teilgraphen notwendig. Während die Rekonfiguration von Prozessdefinitionen zur Modellierungszeit uneingeschränkt durchgeführt werden kann, muss sich die Rekonfiguration zu Laufzeit an dem aktuellen Ausführungsstatus der Prozessinstanz orientieren. Würde die Rekonfiguration zu einer Änderung der Prozesshistorie führen, so käme dies einer Manipulation der medizinischen Dokumentation gleich. Sowohl bei der Ad-hoc-Änderung von Prozessdefinitionen als auch ihrer Instanzen müssen also in Abhängigkeit der gewählten Zielsprache und des Zielsystems, Regeln definiert werden, die die korrekte Rekonfiguration und Transformation des SPF-Graphen sicherstellen. In dieser Arbeit orientierten sich die Regeln an der formalen Definition von Prozessinstanzen auf Basis von ADEPT2 WSM Nets und deren Ausführungssemantik.

Zu Beginn dieses Kapitels wird die Rekonfiguration zur Modellierungszeit behandelt, die vollkommen unabhängig von konkreten Prozessmodellierungssprachen und WfMS stattfinden kann. Es werden mögliche Rekonfigurationsoperationen beschrieben und Bedingungen definiert, die gelten müssen, damit eine solche Operation ausgeführt werden darf. Anschließend wird der grundlegende Unterschied zwischen der strukturerhaltenden und der nicht-strukturerhaltenden Rekonfiguration aufgezeigt. Für beide Methoden werden die entsprechenden Algorithmen spezifiziert. Nachdem dargestellt wurde, wie die Transformation des SPF-Graphen nach seiner Rekonfiguration funktioniert, wird der Ablauf der Rekonfiguration zur Laufzeit nach demselben Schema beschrieben. Da bei diesem Vorgang die Ausführungssemantik der Zielsprache und des Zielsystems berücksichtigt werden müssen, erfolgt zu Beginn dieses Abschnitts eine Einführung in die formalen Grundlagen von Prozessinstanzen von ADEPT2 WSM

Nets. Abschließend wird eine Methode präsentiert, die die Auswahl von Rekonfigurationsknoten im SPF-Graphen für Fachanwender erleichtert und auch bei der Umsetzung von Prozessvarianten, die in Kapitel 4.7 behandelt wird, eine wichtige Rolle spielt.

4.6.1 Rekonfiguration von SPF-Graphen zur Modellierungszeit

Der Bedarf für die Rekonfiguration eines Knoten kann sich schon während der Konfiguration ergeben, wenn die Selektion der Nachfolger eines Knoten nachträglich geändert werden soll. Zur Modellierungszeit beziehen sich die Änderungen am SPF-Graphen also auf die noch nicht instanziierte Prozessdefinition, die im Anschluss an die Rekonfiguration durch Transformation der modifizierten Teilgraphen angepasst werden muss. Algorithmen zur Rekonfiguration von Knoten während der Konfiguration wurden bereits in Kapitel 4.4.3.2 bei der Vorstellung der Lösungskonzepte zur Berücksichtigung von Constraint-Relationen entwickelt. Geht die Verletzung eines Anforderungs-Constraints auf eine fehlerhafte Konfiguration zurück, so wird automatisch der kleinste für eine Rekonfiguration geeignete Knoten identifiziert. Anschließend wird eine temporäre Graphgrammatik generiert, die nur solche Regeln enthält, die die Einhaltung des Constraints bei der Rekonfiguration des Knotens sicherstellen. Gleichzeitig ermöglichen die Algorithmen den Strukturerhalt, also die Beibehaltung bereits existierender, unproblematischer Teilgraphen. Bei einer Verletzung des Ausschluss-Constraints besteht entweder die Option, den identifizierten Teilgraphen durch Entfernen des problematischen Zweigs zu reduzieren oder ihn zu rekonfigurieren. Da Fachanwender bei der Konfiguration nicht unter Zeitdruck stehen, der aktuelle Prozessablauf nicht unterbrochen werden kann und die Rekonfigurationsmöglichkeiten nicht durch den Ausführungsstatus des Prozesses eingeschränkt werden, ähnelt die Rekonfiguration von SPF-Graphen zur Modellierungszeit weitgehend der normalen Konfiguration. So wird z. B. die Einhaltung von Constraint-Relationen vom System weiterhin automatisch erzwungen und muss nicht vom Fachanwender berücksichtigt werden. Dennoch sind auch hier grundlegende Regeln zu beachten, die im folgenden Abschnitt formal spezifiziert werden.

4.6.1.1 Bedingungen für die Rekonfiguration

Zu Beginn der Rekonfigurationsphase muss der Anwender einen Knoten bzw. eine Menge gleichartiger Knoten bestimmen, deren Konfiguration modifiziert werden soll. Wichtig ist an dieser Stelle die Anmerkung, dass sich die Konfiguration nicht auf den selektierten Knoten selbst bezieht, sondern immer nur auf die ihm untergeordneten Nachfolger; der selektierte Knoten hingegen wird auch bei der Rekonfiguration letztlich stets in dasselbe terminale Symbol überführt. Dieser Knoten wird im Folgenden auch als »Rekonfigurationsknoten« bezeichnet. Die einfachste Möglichkeit der Identifikation des Rekonfigurationsknotens besteht in seiner manuellen Selektion. Weitergehende Verfahren zur effizienten Auswahl von Knoten im SPF-Graphen werden in Kapitel 4.6.3 vorgestellt. Grundsätzlich gilt, dass die Rekonfiguration nur für Wurzel- und Kontextknoten möglich ist. Da SPF Knoten keine Nachfolger haben, macht eine Rekonfiguration in ihrem Fall keinen Sinn. Außerdem kommen natürlich nur terminale Knoten für eine Rekonfiguration in Frage, da nicht-terminale Knoten noch normal konfiguriert werden können. Seien $STG = (V^{STG}, E^{STG}, V^{STG}_{SPF}, r^{STG}, ...)$ ein SPF-Typ-Graph und $SG = (V^{SG}, E^{SG}, V^{SG}_{T}, V^{SG}_{NT}, ...)$ ein nach *STG* getypter SPF-Graph. Die Gesamtmenge möglicher Rekonfigurationsknoten in *SG* lässt sich folgendermaßen definieren:

$$V^*_{Reconfig} = \{v \in V^{SG}_{T} \mid g_v(v) \notin V^{STG}_{SPF}\}$$

Nach der Selektion eines oder mehrerer Rekonfigurationsknoten können für den Fachanwender die folgenden Rekonfigurationsoperationen bereit stehen:
- Hinzufügen von Klonknoten
- Hinzufügen von neuen Knoten
- Entfernen von Klonknoten (und die mit ihnen verbundenen Teilgraphen)
- Entfernen von einzelnen Knoten (und die mit ihnen verbundenen Teilgraphen)

Bei Knoten aus der Menge der Rekonfigurationsknoten $V_{Reconfig} \subseteq V^*_{Reconfig}$ kann es sich um Klonknoten oder um unterschiedliche Knoten handeln. In beiden Fällen ist nicht sichergestellt, dass auf jeden Knoten dieser Menge dieselben Rekonfigurationsoperationen angewandt werden können; während Klone unterschiedlich konfiguriert sein können, verweisen unterschiedliche Knoten - außer bei Mehrfachreferenz - auf verschiedene Nachfolgerknoten. Deswegen hängt die Durchführung der Rekonfigurationsoperationen von Bedingungen ab, die für jeden Knoten $v_{Reconfig} \in V_{Reconfig}$ und dessen Nachfolgerknoten individuell geprüft werden müssen. Einflussgrößen sind die aktuelle Konfiguration des Knoten, der zugeordnete logische Operator, die Kardinalitäten der Nachfolger und die Constraint-Relationen zu anderen Knoten. Enthält die Menge $V_{Reconfig}$ mehr als einen Rekonfigurationsknoten und kann eine Operation nicht auf alle Knoten dieser Menge angewendet werden, hat dies die Konsequenz, dass die Rekonfiguration nur für die Knoten der Menge ausgeführt wird, die die entsprechenden Bedingungen erfüllen. Im Folgenden werden die Bedingungen für die Durchführung der vier verschiedenen Rekonfigurationsoperationen angeführt.

Hinzufügen von Klonknoten

Das Hinzufügen eines weiteren Klonknoten ist nur möglich, wenn mindestens ein Knoten dieses Typs als Nachfolger des Rekonfigurationsknotens existiert und noch nicht die maximale Obergrenze der möglichen Klone in *SG* erreicht ist; sei $v^{STG} \in target(g_v(v_{Reconfig}))$ der neu hinzuzufügende Knoten und $GG = (STG^{GG}, PR^{GG}, START^{GG})$ die Graphgrammatik zu *STG*. Dann muss zunächst ermittelt werden, ob es im SPF-Graphen bereits Klone zu diesem Knoten gibt:

$$V^{SG}{}_{Clon} = \{v^{SG} \in target(v_{Reconfig}) \mid g_v(v^{SG}) = v^{STG} \vee (\exists (v^{STG}, L, R) \in PR^{GG}: L = (v^{SG}, \varnothing, \varnothing, v^{SG}, v^{SG}, A^{SG}, VA^{SG}, VID^{SG}))\}$$

V_{Clon} kann demnach sowohl terminale als auch nicht-terminale Knoten enthalten, die Klonen von v^{SG} entsprechen. Dann stellt sich die Bedingung für das Hinzufügen von Klonknoten zu v^{SG} folgendermaßen dar:

$$|V^{SG}{}_{Clon}| > 0 \wedge (VO^{STG}(v^{STG}) - |V^{SG}{}_{Clon}|) > 0$$

Diese Bedingung sagt aus, dass die Menge V_{Clon} mindestens einen Knoten umfassen muss und die maximale Knotenkardinalität größer ist als die aktuelle Anzahl von Klonen im SPF-Graphen. Prinzipiell besteht bei allen Operatoren die Möglichkeit, Klone von bereits vorhandenen Nachfolgerknoten einzufügen, sofern die oben spezifizierten Bedingungen erfüllt sind.

Hinzufügen von neuen Knoten

Das Hinzufügen von komplett neuen Knoten ist bei Rekonfigurationsknoten mit AND-Operator grundsätzlich nicht möglich. Bei Knoten mit XOR-Operator besteht die Option nur dann, wenn alle vorher existierenden Nachfolgerknoten entfernt wurden. Diesen Zustand kann der Rekonfigurationsknoten jedoch nur in nicht-terminaler Form annehmen, da er als terminaler Knoten immer mindestens einen Kindknoten besitzen muss; d. h. die Rekonfiguration eines Knotens mit XOR-Operator kommt immer dem Ersetzen des Knotens und des assoziierten Teilgra-

phens durch sein Nichtterminalsymbol gleich. Bei Knoten mit OR- und OPT-Operator können neue Knoten hinzugefügt werden, solange es im SPF-Typ-Graphen Nachfolger gibt, die noch nicht im SPF-Graphen unterhalb von $v_{Reconfig}$ vorkommen.

$$VO^{STG}(g_V(v_{Reconfig})) \in \{OR, OPT\} \Rightarrow \exists v^{STG} \in target(g_V(v_{Reconfig})) \nexists v^{SG} \in target(v_{Reconfig}): g_V(v^{SG}) = v^{STG} \vee (\exists (v^{STG}, L, R) \in PR^{GG}: L = (v^{SG}, \varnothing, \varnothing, v^{SG}, v^{SG}, A^{SG}, VA^{SG}, VID^{SG}))$$

Dementsprechend ist die Menge von Knoten im SPF-Typ-Graphen, die unterhalb von $v_{Reconfig}$ in den SPF-Graphen eingefügt werden können, folgendermaßen definiert:

$$V^{STG}_{Add} = \{v^{STG} \in V^{STG} | v^{STG} \in target(g_V(v_{Reconfig})) \wedge (\nexists v^{SG} \in target(v_{Reconfig}): g_V(v^{SG}) = v^{STG} \vee (\exists (v^{STG}, L, R) \in PR^{GG}: L = (v^{SG}, \varnothing, \varnothing, v^{SG}, v^{SG}, A^{SG}, VA^{SG}, VID^{SG}))$$

Gemäß Definition existiert für die Knoten dieser Menge im SPF-Graphen bisher kein Abbild weder in Form eines terminalen noch in Form eines nicht-terminalen Knotens. Zu beachten ist, dass das Hinzufügen neuer Knoten im Gegensatz zu dem Hinzufügen von Klonknoten zur Verletzung von Ausschluss- und Anforderungs-Constraints führen kann. Ist der neu eingefügte Knoten über einen Ausschluss-Constraint mit einem anderen Knoten verbunden, so muss dieser zweite Knoten wieder aus dem SPF-Graphen entfernt werden. Gibt es hingegen einen Anforderungs-Constraint zu einem Knoten, der noch nicht im SPF-Graphen existiert, so muss auch dieser zweite Knoten, je nach der aktuellen Struktur des SPF-Graphens, durch Konfiguration oder Rekonfiguration ebenfalls hinzugefügt werden. Die Einhaltung beider Constraint-Relationen lässt sich zur Modellierungszeit durch die in Kapitel 4.4.3.2 spezifizierten Algorithmen gewährleisten.

Entfernen von Klonknoten

Ob das Entfernen eines Knoten und des mit ihm assoziierten Teilgraphen überhaupt möglich ist, hängt von dem logischen Operator ab, der dem Rekonfigurationsknoten zugeordnet ist. Das Entfernen von Klonen von bereits vorhandenen Knoten ist jedoch bei allen logischen Verknüpfungen möglich. Es muss dann allerdings auch sichergestellt sein, dass mindestens ein Knoten dieses Typs weiterhin unterhalb des Rekonfigurationsknotens bestehen bleibt. Sei $v^{SG} \in target(v_{Reconfig})$ ein Nachfolgerknoten des Rekonfigurationsknotens, der gelöscht werden soll. Zunächst muss wie oben überprüft werden, ob es sich um einen Klonknoten handelt. Die Menge V^{SG}_{Clon} wird nun ausgehend von dem zu entfernenden Knoten v^{SG} definiert:

$$V^{SG}_{Clon} = \{u^{SG} \in target(v_{Reconfig}) | u^{SG} \neq v^{SG} \wedge (g_V(u^{SG}) = g_V(v^{SG}) \vee (\exists v^{STG} \in V^{STG}: (v^{STG} = g_V(v^{SG}) \wedge \exists (v^{STG}, L, R) \in PR^{GG}: L = (u^{SG}, \varnothing, \varnothing, u^{SG}, u^{SG}, A^{SG}, VA^{SG}, VID^{SG})) \vee (v^{STG} = g_V(u^{SG}) \wedge \exists (v^{STG}, L, R) \in PR^{GG}: L = (v^{SG}, \varnothing, \varnothing, v^{SG}, v^{SG}, A^{SG}, VA^{SG}, VID^{SG})) \vee (\exists (v^{STG}, L, R) \in PR^{GG}: L = (u^{SG}, \varnothing, \varnothing, u^{SG}, u^{SG}, A^{SG}, VA^{SG}, VID^{SG}) \wedge \exists (v^{STG}, L, R) \in PR^{GG}: L = (v^{SG}, \varnothing, \varnothing, v^{SG}, v^{SG}, A^{SG}, VA^{SG}, VID^{SG})))\}$$

Insgesamt müssen bei der Ermittlung der Menge V^{SG}_{Clon} vier Fälle unterschieden werden:

- Es handelt sich bei v^{SG} und u^{SG} jeweils um terminale Knoten,
- v^{SG} ist ein terminaler Knoten, u^{SG} ein nicht-terminaler Knoten,
- v^{SG} ist ein nicht-terminaler Knoten, u^{SG} ein terminaler Knoten und
- sowohl v^{SG} als auch u^{SG} sind nicht-terminale Knoten.

Damit der Knoten als Klonknoten aus dem SPF-Graphen entfernt werden kann, muss die Menge V^{SG}_{Clon} über mindestens ein Element verfügen:

$$|V^{SG}_{Clon}| > 0$$

Entfernen von einzelnen Knoten

Das Entfernen eines einzelnen Knotens und dessen Teilgraphen erfolgt immer dann, wenn es zu diesem Knoten keine Klone unterhalb des Rekonfigurationsknotens gibt. Sei V_{Clon} wie oben definiert, dann gilt:

$$|V^{SG}_{Clon}| = 0$$

Das Entfernen von Knoten und Teilgraphen ist generell nicht möglich, wenn dem zu rekonfigurierenden Knoten ein AND-Operator zugeordnet ist. In diesem Fall gibt es also ausschließlich die Möglichkeit, Klone zu den bereits existierenden Knoten einzufügen oder wieder zu entfernen; auch das Löschen von Knoten unterhalb eines Rekonfigurationsknoten mit XOR-Operator ist nicht ohne weiteres möglich, da dieser mindestens einen Nachfolger benötigt, der Abbild genau eines Knotens im SPF-Typ-Graphen ist. Daher folgt als Bedingung für das Entfernen einzelner Knoten und Teilgraphen für den Rekonfigurationsknoten:

$$VO^{STG}(g_V(v_{Reconfig})) \in \{OR, OPT\}$$

Bei Rekonfigurationsknoten mit OR-Operator können neben Klonknoten auch alle anderen Nachfolgerknoten aus dem SPF-Graph entfernt werden; es gilt lediglich die Bedingung, dass nach dem Entfernen von Nachfolgerknoten mindestens noch ein Knoten unterhalb des Rekonfigurationsknoten vorhanden ist:

$$VO^{STG}(g_V(v_{Reconfig})) = OR \Rightarrow target(v_{Reconfig}) > 1$$

Im Hinblick auf den OPT-Operator sind dem Löschen von Knoten hingegen keinerlei Grenzen gesetzt, da Knoten mit OPT-Operator nicht zwingend Nachfolger besitzen müssen. Ähnlich wie beim Hinzufügen neuer Knoten, kann es auch beim Entfernen von Knoten vorkommen, dass ein Anforderungs-Constraint verletzt wird. Erfolgt die Rekonfiguration jedoch zur Modellierungszeit, so kann die Einhaltung dieses Constraints wiederum nachträglich erzwungen werden. Sind die obigen Bedingungen erfüllt, dann entspricht die Menge der Knoten im SPF-Graphen, die entfernt werden können, genau der Menge der Nachfolgerknoten von $v_{Reconfig}$ für die keine Klone existieren:

$$V^{SG}_{Del} = \{v^{SG} \in target(v_{Reconfig}) | \nexists\, u^{SG} \in target(v_{Reconfig}): u^{SG} \neq v^{SG} \wedge (g_V(u^{SG}) = g_V(v^{SG}) \vee (\exists v^{STG} \in V^{STG}: (g_V(v^{SG}) = v^{STG} \wedge \exists\, (v^{STG}, L, R) \in PR^{GG}: L = (u^{SG}, \varnothing, \varnothing, u^{SG}, u^{SG}, A^{SG}, VA^{SG}, VID^{SG})) \vee (g_V(u^{SG}) = v^{STG} \wedge \exists\, (v^{STG}, L, R) \in PR^{GG}: L = (v^{SG}, \varnothing, \varnothing, v^{SG}, v^{SG}, A^{SG}, VA^{SG}, VID^{SG})) \vee (\exists (v^{STG}, L, R) \in PR^{GG}: L = (u^{SG}, \varnothing, \varnothing, u^{SG}, u^{SG}, A^{SG}, VA^{SG}, VID^{SG}) \wedge \exists\, (v^{STG}, L, R) \in PR^{GG}: L = (v^{SG}, \varnothing, \varnothing, v^{SG}, v^{SG}, A^{SG}, VA^{SG}, VID^{SG}))))\}$$

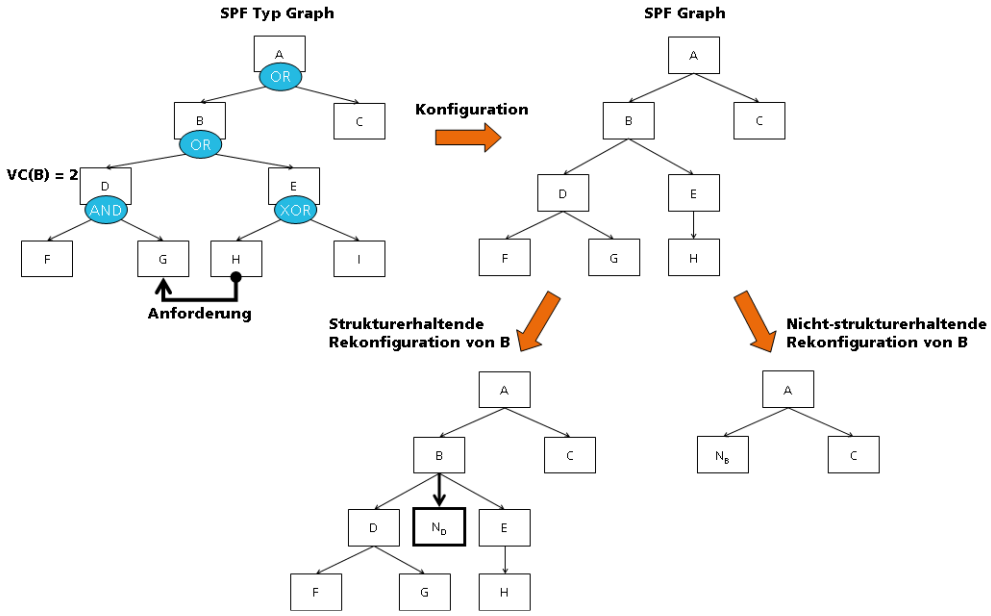
Wie bei der obigen Definition der Menge über alle Klone muss auch hier in Betracht gezogen werden, dass es sich bei v^{SG} und u^{SG} jeweils um terminale oder nicht-terminale Knoten handeln kann.

4.6.1.2 Strukturerhaltende und nicht-strukturerhaltende Rekonfiguration

Bei der Rekonfiguration eines Knotens kann grundsätzlich zwischen einer strukturerhaltenden und einer nicht-strukturerhaltenden Vorgehensweise unterschieden werden. Die strukturerhaltende Rekonfiguration zielt auf den Erhalt bereits existierender Teilgraphen unterhalb des Rekonfigurationsknotens ab. Die nicht strukturerhaltende Rekonfiguration bedeutet hingegen das Zurücksetzen des Rekonfigurationsknotens auf das entsprechende Nichtterminalsymbol und die erneute Konfiguration des Knotens. Diese radikale Methode maximiert zwar die Freiheiten bei der Rekonfiguration, sie ist aber nicht immer erwünscht; insbesondere dann nicht, wenn

der betroffene Teilgraph über eine Vielzahl an Nachfolgern verfügt und seine erneute Konfiguration zwar nicht erforderlich aber mit einem hohen Arbeitsaufwand verbunden ist. Abbildung 101 verdeutlicht an einem Beispiel den Unterschied zwischen der strukturerhaltenden und der nicht-strukturerhaltenden Rekonfiguration beim Einfügen eines Klonknotens.

Abbildung 101 Unterschied zwischen strukturerhaltender und nicht-strukturerhaltender Rekonfiguration von SPF-Graphen



Wie das Beispiel oben zeigt, soll unterhalb von Knoten *B* ein neuer Klon von *D* eingefügt werden. Bei der strukturerhaltenden Rekonfiguration wird der bereits existierende Teilgraph nicht verändert; mit dem nicht-terminalen Knoten N_D steht lediglich ein konfigurierbarer Klon von *D* zur Verfügung. Bei der nicht-strukturerhaltenden Rekonfiguration von *B* wird hingegen der gesamte Teilgraph unterhalb von *B* aus dem SPF-Graphen entfernt.

Die nicht-strukturerhaltende Rekonfiguration stellt die einfachere Variante der beiden Rekonfigurationsverfahren dar und kann unabhängig von den in Kapitel 4.6.1.1 geltenden Bedingungen durchgeführt werden. Sie beinhaltet lediglich das Ersetzen des Rekonfigurationsknoten durch das entsprechende Nichtterminalsymbol. Die Rekonfiguration des Knoten kann anschließend durch automatisierte Konfiguration (siehe Kapitel 4.4.4.1) und Evaluation von Constraint-Relationen (siehe Kapitel 4.4.3.2) beeinflusst werden. Zur Modellierungszeit ist die nicht-strukturerhaltende Rekonfiguration von Knoten des SPF-Graphen uneingeschränkt möglich. Die folgende Funktion entspricht dem Algorithmus für die nicht-strukturerhaltende Rekonfiguration. Diese erhält als Parameter den SPF-Typ-Graphen *STG*, den SPF-Graphen *SG* und den Rekonfigurationsknoten $v_{Reconfig}$.

Formel 18 Algorithmus zur nicht-strukturerhaltenden Rekonfiguration einer Knotenmenge zur Modellierungszeit

$$
\begin{array}{l}
\textit{boolean reconfigureNodeBuildTime}(STG, SG, v_{Reconfig}) \{ \\
\quad \text{IF } (v_{Reconfig} \in V^{SG}_{T} \wedge g_v(v_{Reconfig}) \notin V^{STG}_{SPF}) \\
\quad\quad \text{sourceNode}^{SG} := \{v \in V^{SG} | v \rightarrow v_{Reconfig}\};
\end{array}
$$

```
        succNodes^SG := {v ∈ V^SG|v_Reconfig →* v};
        E^SG := E^SG - {(x^SG, y^SG) ∈ E^SG|x ∈ (sourceNode^SG ∪ v_Reconfig ∪ succNodes^SG)
∧ y^SG ∈ (v_Reconfig ∪ succNodes^SG)};
        V^SG := V^SG - (v_Reconfig ∪ succNodes^SG);
        V^SG := V^SG ∪ N_vReconfig;
        E^SG := E^SG ∪ (sourceNode^SG, N_vReconfig);
    END IF
}
```

Nach der Ermittlung von Vorgänger- und Nachfolgerknoten des aktuellen Rekonfigurationsknotens, werden zunächst die Kanten- und anschließend die Knotenmenge des Teilgraphens mit Wurzel $v_{Reconfig}$ aus dem SPF-Graphen entfernt. Anschließend wird ein neuer nicht-terminaler Knoten für $v_{Reconfig}$ in den SPF-Graphen eingefügt und über eine gerichtete Kante mit dem entsprechenden Vorgängerknoten verbunden. Nach dem Ersetzen der Teilgraphen durch das Nichtterminalsymbol des jeweiligen Wurzelknotens können Fachanwender mit der erneuten Konfiguration des Knotens beginnen.

Die strukturerhaltende Rekonfiguration kann nur in Abhängigkeit der gewünschten Operation, der logischen Verknüpfungen, Knotenkardinalitäten und Constraint-Relationen durchgeführt werden. Sie kann generell beim Hinzufügen von Klonknoten angewandt werden und sollte in diesem Fall auch das Verfahren der Wahl darstellen. Beim Einfügen neuer Knoten ist sie nicht möglich, wenn der Rekonfigurationsknoten mit einem XOR-Operator assoziiert ist; in diesem Fall bedeutet das Hinzufügen eines neuen Knotens immer das Entfernen der gesamten, bisherigen Struktur, was einer nicht-strukturerhaltenden Rekonfiguration entspricht. Das Entfernen der Nachfolger unterhalb eines Rekonfigurationsknotens mit AND-Operator oder das Löschen aller Nachfolger bei Knoten mit OR-Operator ist generell nicht möglich. Bevor die strukturerhaltende Rekonfiguration ausgeführt wird, müssen also die in Kapitel 4.6.1.1 spezifizierten Bedingungen überprüft werden. In Abhängigkeit der Evaluationsergebnisse kann den Fachanwendern dann auch eine strukturerhaltende Rekonfiguration angeboten werden. Die nächste Funktion entspricht dem Algorithmus zum Hinzufügen eines nicht-terminalen Knotens entweder als Klon oder als neuer Knoten bei Strukturerhalt; sie erhält als Parameter den SPF-Typ-Graphen, den daraus abgeleiteten SPF-Graphen, den Rekonfigurationsknoten und die Bezeichnung des Knotens aus dem SPF-Typ-Graphen, für den ein Klonknoten oder ein neuer Knoten im SPF-Graph erzeugt werden soll.

Formel 19 Algorithmus für das strukturerhaltende Einfügen nicht-terminaler Knoten in SPF-Graphen zur Modellierungszeit

```
addNodeBuildTime(STG, SG, v_Reconfig, v^STG_Add) {
    IF(v_Reconfig ∈ V^SG)
        IF (v^STG_Add ∈ target(g_v(v_Reconfig)))
            maxCardinality := VC^STG(v^STG_Add);
            V_Clon := {v^SG ∈ target(v_Reconfig)|g_v(v^SG) = v^STG_Add ∨ (∃(v^STG_Add, L, R) ∈
PR^GG: L = (v^SG, ∅ , ∅ , v^SG, v^SG, A^SG, VA^SG, VID^SG))};
            IF (|V_Clon| > 0 ∧ maxCardinality - |V_Clon| > 0)
                //Einfügen des Nichtterminalknotens als Klonknoten
                V^SG := V^SG ∪ N^1_vSTGAdd;
                E^SG := E^SG ∪ (v_Reconfig, N^1_vSTGAdd);
            END IF
            ELSE IF (|V_Clon| = 0 ∧ VO^STG(g_v(v_Reconfig)) ∈ {OR, OPT}) {
                //Einfügen des Nichtterminalknotens als neuen Knoten
                V^SG := V^SG ∪ N_vSTGAdd;
                E^SG := E^SG ∪ (v_Reconfig, N_vSTGAdd);
            END ELSE IF
```

```
        END IF
      END IF
}
```

Zuerst wird kontrolliert ob der Rekonfigurationsknoten im SPF-Graphen existiert und ob es sich bei v^{STG}_{Add} gemäß SPF-Typ-Graphen wirklich um seinen Nachfolgerknoten handelt. Ist dies der Fall, wird die Bedingung für das Einfügen eines Klonknotens überprüft. Die erste Anforderung ist, dass der Rekonfigurationsknoten bereits über mindestens einen terminalen oder nicht-terminalen Nachfolgerknoten dieses Typs verfügt; die zweite Anforderung besteht darin, dass die Anzahl der Klone unterhalb der maximalen Knotenkardinalität im SPF-Typ-Graphen liegen muss. Sind diese Bedingungen erfüllt, wird das Nichtterminalsymbol des Klons als neuer Knoten unterhalb des Rekonfigurationsknotens eingefügt. Gibt es hingegen noch keinen gleichartigen Knoten und handelt es sich bei dem logischen Operator des Rekonfigurationsknotens um eine OR- oder OPT-Verknüpfung, so wird das Nichtterminalsymbol von v^{STG}_{Add} als neuer Knoten dem SPF-Graphen hinzugefügt.

Aus dieser Funktion wird ersichtlich, dass die Anwendung der Produktionsregeln bei der strukturerhaltenden Rekonfiguration umgangen wird. Eine Alternative würde in der Erstellung einer temporären Grammatik für jeden Rekonfigurationsknoten bestehen, der analog zu den Algorithmen aus Kapitel 4.4.3.2 die bestehenden Strukturen berücksichtigt. Je nachdem wie deterministisch diese Grammatik erzeugt wird, könnte der Fachanwender die zuvor vorhandene Struktur während der Rekonfiguration beeinflussen oder sie würde mit Hilfe der automatisierten Konfiguration ohne menschliche Interaktion erstellt werden. Eine temporäre Grammatik, die Fachexperten Optionen hinsichtlich der wiederholten Konfiguration bereits existierender Strukturen ermöglicht, erscheint jedoch nicht zielführend im Fall der Rekonfiguration; Hintergrund der Rekonfiguration ist ein spontaner Änderungswunsch, der rasch und mit minimalem Aufwand umgesetzt werden soll. Eine vollkommen deterministische Grammatik hingegen würde ein exaktes Abbild der bereits zuvor vorhandenen Struktur erstellen; der dafür notwendige Rechenaufwand kann angesichts des Ergebnisses kaum gerechtfertigt sein. Da der oben angeführte Algorithmus SPF-Graphen nur dann verändert, wenn die Rekonfigurationsbedingungen für das Hinzufügen von Knoten erfüllt sind, kann die Konformität des SPF-Graphen mit dem SPF-Typ-Graphen weiterhin gewährleistet werden. Der Einsatz von Graphgrammatiken dient in erster Linie der Sicherstellung der Korrektheit eines SPF-Graphen, wenn die Konfiguration manuell durch Fachanwender erfolgt.

Die nachfolgende Funktion realisiert einen Algorithmus zur Entfernung von Knoten und den mit ihnen assoziierten Teilgraphen unterhalb eines Rekonfigurationsknotens unter Beibehaltung aller übrigen Strukturen. Dabei bestimmt der Fachanwender zuerst den zu entfernenden Knoten bzw. Graphen und führt anschließend die Löschoperation durch. Der Funktion *deleteNodeBuildTime* wird neben dem SPF-Typ-Graphen und dem SPF-Graphen ein Rekonfigurationsknoten $v_{Reconfig}$ übergeben sowie die eindeutige Bezeichnung v^{SG}_{Del} des Wurzelknotens des zu löschenden Teilbaums im SPF-Graphen.

Formel 20 Algorithmus für das strukturerhaltende Entfernen von Teilgraphen aus SPF-Graphen zur Modellierungszeit

$$
\begin{aligned}
&\textit{deleteNodeBuildTime}(STG, SG, v_{Reconfig}, v^{SG}_{Del}) \{\\
&\quad IF\ (v^{SG}_{Del} \in target(v_{Reconfig}))\\
&\qquad V_{Clon} \coloneqq \{v^{SG} \in target(v_{Reconfig}) \mid v^{SG} \neq v^{SG}_{Del} \wedge (g_v(v^{SG}) = g_v(v^{SG}_{Del}) \vee (\exists v^{STG}\\
&\in V^{STG}: (v^{STG} = g_v(v^{SG}_{Del}) \wedge \exists (v^{STG}, L, R) \in PR^{GG}: L = (v^{SG}, \varnothing, \varnothing, v^{SG}, v^{SG}, VA,
\end{aligned}
$$

```
VID)) ∨ (v^STG = g_v(v^SG) ∧ ∃(v^STG, L, R) ∈ PR^GG: L = (v^SG_Del, ∅ , ∅ , v^SG_Del, v^SG_Del, VA,
VID)) ∨ (∃(v^STG, L, R) ∈ PR^GG: L = (v^SG, ∅ , ∅ , v^SG, v^SG, VA, VID) ∧ ∃(v^STG, L,
R) ∈ PR^GG: L = (v^SG_Del, ∅ , ∅ , v^SG_Del, v^SG_Del, VA, VID)))};
         IF (|V_Clon| > 0)
            //Strukturerhaltendes Entfernen des Klonknoten
            V_Del := {v ∈ V^SG|v^SG_Del →* v} ∪ v^SG_Del;
            E_Del := {(x, y) ∈ E^SG|x, y ∈ V_Del} ∪ (v_Reconfig, v^SG_Del);
            E^SG := E^SG - E_Del;
            V^SG := V^SG - V_Del;
         END IF
         ELSE IF (|V_Clon| = 0 ∧ VO^STG(g_v(v_Reconfig)) ≠ AND)
            IF (VO^STG(g_v(v_Reconfig)) = OPT ∨ (VO^STG(g_v(v_Reconfig)) = OR ∧
|target(v_Reconfig)| > 1))
               //Strukturerhaltendes Entfernen des Knoten
               V_Del := {v ∈ V^SG|v^SG_Del →* v} ∪ v^SG_Del;
               E_Del := {(x, y) ∈ E^SG|x, y ∈ V_Del} ∪ (v_Reconfig, v^SG_Del);
               E^SG := E^SG - E_Del;
               V^SG := V^SG - V_Del;
            END IF
            ELSE
               //Nicht-strukturerhaltende Rekonfiguration bei XOR- oder OR-
Operator ohne weitere Nachfolger
               reconfigureNodeBuildTime(SG, v_Reconfig);
               return true;
            END ELSE
         END ELSE IF
      END IF
}
```

Eine grundlegende Bedingung für das Entfernen eines Knotens und des assoziierten Teilgraphen ist die Existenz des entsprechenden Knotens unterhalb des Rekonfigurationsknotens im SPF-Graphen. Anschließend wird die Menge aller terminalen und nicht-terminalen Klone dieses Knotens bestimmt. Enthält diese Menge mehr als ein Element, so entspricht die Operation dem Entfernen eines Klonknotens und darf prinzipiell unabhängig von dem logischen Operator, der dem Rekonfigurationsknoten zugeordnet ist, ausgeführt werden. Gibt es hingegen keine Klone von dem zu löschenden Knoten unterhalb des Rekonfigurationsknotens, so darf dem Rekonfigurationsknoten kein AND-Operator zugeordnet sein. Trifft dies zu, so kann der Knoten strukturerhaltend entfernt werden, sofern dem Rekonfigurationsknoten ein OPT- oder ein OR-Operator zugeordnet ist; bei einem OR-Operator gilt zusätzlich die Bedingung, dass der Rekonfigurationsknoten nach dem Löschvorgang noch mindestens einen Nachfolgerknoten umfasst. Ist diese Bedingung nicht erfüllt oder verfügt der Rekonfigurationsknoten über einen XOR-Operator, so bleibt nur die Möglichkeit der nicht-strukturerhaltenden Rekonfiguration.

4.6.1.3 Transformation von SPF-Graphen nach der Rekonfiguration

Da SPF-Graphen nur auf zwei Arten verändert werden können, nämlich durch das Entfernen existierender Knoten oder das Hinzufügen von neuen Knoten, werden die Auswirkungen der Änderungsoperationen auf die Prozessdefinition bzw. deren Instanz durch die Regeln der in Kapitel 4.5.3 spezifizierten Transformationsphase determiniert; die für die Anwender sichtbaren Auswirkungen können dabei nicht nur dem Einfügen oder Löschen von Prozessaktivitäten entsprechen, sondern z. B. auch ihrem Verschieben oder Ersetzen. In jedem Fall bilden die überschaubaren Manipulationsmöglichkeiten am SPF-Graphen die Basis für die Umsetzung sowohl einfacher als auch komplexer Modifikationen an der Prozessdefinition.

Die simpelste Möglichkeit die Änderungen am SPF-Graphen in die Prozessdefinition zu übernehmen, besteht in der erneuten Transformation des gesamten Graphen. Da dies insbesondere bei kleinen Änderungen an umfangreichen SPF-Graphen zu einer schlechten Performanz führen würde, besteht ein effizienteres Vorgehen darin, die Transformation nur auf die neu eingefügten Knoten zu beziehen. Das Löschen von Knoten oder Teilgraphen aus dem SPF-Graphen muss in diesem Fall allerdings separat auch für die betroffenen Prozessfragmente in der Prozessdefinition durchgeführt werden. Sei $STG = (V^{STG}, E^{STG}, ...)$ ein SPF-Typ-Graph, $SG = (V^{SG}, E^{SG}, ...)$ sein SPF-Graph und $W = (N^W, D^W, ...)$ die zugehörige Prozessdefinition. Die Menge der zu löschenden Prozessaktivitäten entspricht der Menge der Prozessfragmente, für die nach der Rekonfiguration keine Knoten mehr im SPF-Graphen existieren:

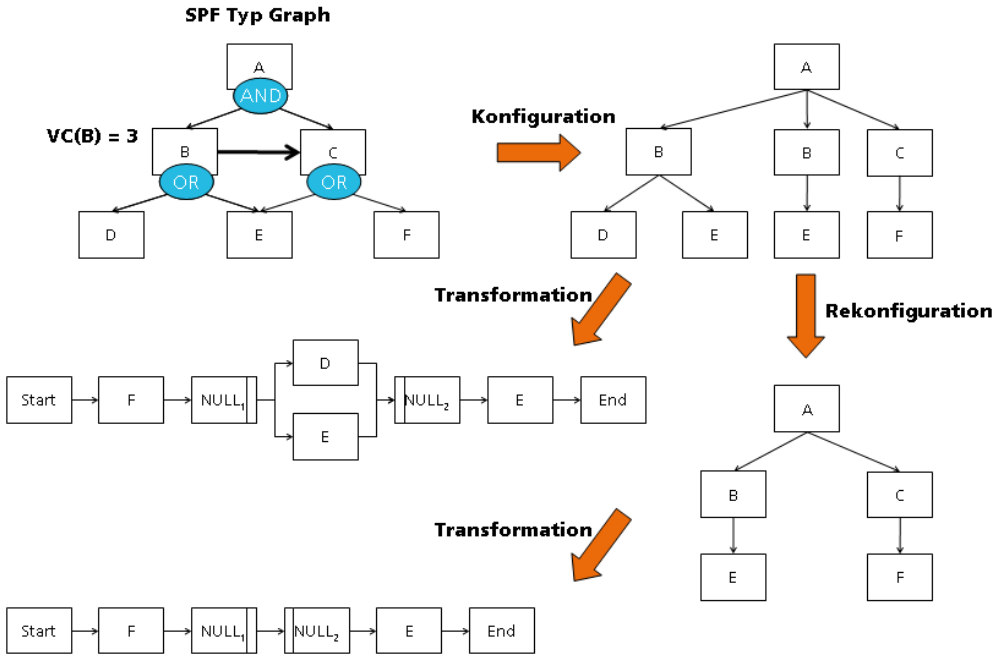
$$N^W_{Delete} = \{n \in N^W | NT^W(n) \notin \{StartFlow, EndFlow, AndSplit, AndJoin\} \wedge \nexists\ v^{SG} \in V^{SG}: n = Name^p \cup VID^{SG}(v^{SG}) \text{ mit } p = VP^{STG}(g_v(v^{SG}))\}$$

Im Fall einer Transformation der SPF-Graphen in ADEPT2 WSM Nets muss anschließend für jede dieser Aktivitäten die Funktion *deleteActivity* aufgerufen werden, die bereits in Kapitel 4.5.2 vorgestellt wurde:

$$\forall n \in N^W_{Delete} \Rightarrow deleteActivity(W, n)$$

Bei Nutzung von ADEPT2 genügt der Aufruf dieser Funktion, um auch alle Kontrollfluss-, Datenfluss- und Synchronisationskanten zu entfernen, die mit dieser Aktivität verknüpft sind. Wird hingegen die Transformation von SPF-Graphen in andere Prozessmodellierungssprachen angestrebt, so müssen gegebenenfalls neben den Aktivitäten auch die entsprechenden Kanten aus der Prozessdefinition entfernt werden. Die folgende Abbildung verdeutlicht Ablauf und Ergebnis der erneuten Transformation nach Rekonfiguration eines SPF-Graphen.

Abbildung 102 Löschen von Prozessaktivitäten nach der Rekonfiguration des SPF-Graphen



Obwohl die parallelen Verzweigungsknoten $NULL_1$ und $NULL_2$ nach der Transformation nicht mehr benötigt werden, werden solche Strukturen von ADEPT2 nicht automatisch entfernt.

Dies ist jedoch möglich, wenn nach jedem Transformationsvorgang in Folge einer Rekonfiguration, das WSM Net gezielt nach solchen Strukturen durchsucht wird; bei Auffinden zwei direkt aufeinander folgender Knoten vom Typ *AndSplit* und *AndJoin* wird dann die in Kapitel 4.5.2 vorgestellte Änderungsoperation *deleteBlock* aufgerufen:

$$\forall n \in N^W: NT^W(n) = AndSplit \wedge NT^W(succ(n)) = AndJoin \Rightarrow deleteBlock(W, n, succ(n))$$

Beim Einfügen neuer Knoten in den SPF-Graphen, wird analog vorgegangen. Zunächst wird nach der Rekonfiguration die Menge aller terminalen Blattknoten im SPF-Graphen ermittelt, für die es im WSM Net noch keine Prozessaktivität gibt. Anschließend folgt das Einfügen nach Rekonfiguration den Regeln der Transformationsphase aus Kapitel 4.5.3, wobei die Menge der zu transformierenden Knoten V_{Trans} nun nicht mehr aus allen terminalen Blattknoten des SPF-Graphen, sondern nur aus den nachträglich hinzugefügten Knoten besteht:

$$V_{Trans} = \{v^{SG} \in V^{SG}{}_T | g_V(v^{SG}) \in V^{STG}{}_{SPF} \wedge \nexists\, n \in N^W: n = Name^p \cup VID^{SG}(v^{SG}) \text{ mit } p = VP^{STG}(g_V(v^{SG}))\}$$

Wie schon bei der Transformationsphase, wird auch hier davon ausgegangen, dass es sich zum Zeitpunkt der Transformation um einen vollständig konfigurierten SPF-Typ-Graphen handelt; Ansätze zum geeigneten Umgang mit unvollständigen Konfigurationen wurden in Kapitel 4.5.5 vorgestellt. Wurden im Zuge der Rekonfiguration sowohl Knoten aus dem SPF-Graphen entfernt als auch neue hinzugefügt, sollte der Löschvorgang stets dem Einfügevorgang vorausgehen, da so die unnötige Berücksichtigung von Abhängigkeiten zu Löschkandidaten vermieden wird.

4.6.2 Rekonfiguration von SPF-Graphen zur Laufzeit

Während Fachanwendern bei der Rekonfiguration zur Modellierungszeit prinzipiell dieselben Möglichkeiten zur Verfügung stehen wie bei der Konfiguration, werden die Freiheiten bei der Rekonfiguration zur Laufzeit durch den aktuellen Ausführungsstatus der betroffenen Prozessinstanz determiniert. Das hat zur Folge, dass z. B. ein Knoten aus dem SPF-Graphen nicht entfernt werden darf, weil mit der Ausführung des mit ihm assoziierten Prozessfragments bereits begonnen wurde oder die Ausführung sogar schon abgeschlossen ist. Diese Anforderung gewinnt vor allem im Hinblick auf Constraint-Relationen an Komplexität, da das Einfügen eines Knotens in den SPF-Graphen auch dadurch verhindert werden kann, dass eine andere Aktivität bereits ausgeführt wurde, zu der es im SPF-Typ-Graphen einen Ausschluss-Constraint gibt. Während die automatische Überprüfung und Erzwingung der Einhaltung von Constraint-Relationen zur Modellierungszeit den Arbeitsaufwand von Fachanwendern reduziert, kann sie speziell bei der Rekonfiguration von Prozessinstanzen aber zu unerwünschten Effekten führen. Angenommen, eine Anwenderin hat einen Knoten derart rekonfiguriert, dass dieser nun einen Nachfolgerknoten besitzt, der in Ausschlussbeziehung zu einem detaillierten Teilbaum im SPF-Graphen steht. Die Nicht-Beachtung dieses Constraints führt nun dazu, dass dieser Teilbaum noch einmal neu konfiguriert werden muss. In der Prozessdefinition wirkt sich dies nach erneuter Transformation so aus, dass alle geplanten Prozessaktivitäten verschwinden und gegebenenfalls durch eine komplexe Verzweigungsstruktur oder eine noch zu konfigurierende Platzhalteraktivität ersetzt werden. Diese Effekte sind zumal unter dem Aspekt, dass an der Patientenbehandlung viele unterschiedliche Personen beteiligt sind, deren Arbeiten vorausschauend aufeinander abgestimmt werden müssen, häufig nicht tolerierbar. Anders als bei der Rekonfiguration von SPF-Graphen zur Modellierungszeit, werden bei der Rekonfigura-

tion zur Laufzeit also teilweise andere Methoden benötigt, die an die veränderten Rahmenbedingungen, die für Prozessinstanzen gelten, angepasst sind.

Da bei der Rekonfiguration von Instanzen die Ausführungssemantik eines WfMS berücksichtigt werden muss, kann diese Phase nicht unabhängig von speziellen Systemen und Prozessmodellierungssprachen erörtert werden; auch die Ad-hoc-Änderung von Prozessinstanzen wird daher am Beispiel von ADEPT2 WSM Nets und deren Ausführungssemantik behandelt. Zu diesem Zweck wird zunächst die Ausführungssemantik von WSM Nets dargestellt, sofern sie für die Rekonfiguration von SPF-Graphen von Bedeutung ist. Anschließend kann dann gezeigt werden, welche erweiterten Bedingungen sich bei der Rekonfiguration von Prozessinstanzen in Abhängigkeit der bisherigen Prozesshistorie ergeben. Zusätzlich zu den in Kapitel 4.6.1.1 dargestellten Bedingungen werden Kriterien zur adäquaten Beachtung von Anforderungs- und Ausschluss-Constraints spezifiziert. Das Kapitel schließt mit der Spezifikation der Transformation von Prozessinstanzen nach der Rekonfiguration.

4.6.2.1 Prozessinstanzen auf Basis von WSM Nets und deren Ausführungssemantik

Ausgehend von der Definition von WSM Nets in Kapitel 4.5.1 werden in diesem Abschnitt die Eigenschaften der Prozessinstanzen spezifiziert, die auf Basis von WSM Net Prozessdefinitionen gebildet werden können. Sämtliche in diesem Abschnitt angegebenen Definitionen basieren auf den Vorarbeiten von Reichert und Rinderle [Reichert 2000b, Rinderle 2004]. Sei $W = (N^W, D^W, NT^W, ...)$ ein korrektes WSM Net, von dem Prozessinstanzen erzeugt und gestartet werden können. Jeder Instanz I ist dabei eine Kopie der instanz-spezifischen Prozessdefinition zugeordnet $W_I = W + \Delta_I$; wobei Δ_I bei Instanzen, deren Prozessdefinition gegenüber dem Original nicht geändert wurde, der leeren Menge entspricht. Im Gegensatz zu Prozessdefinitionen verfügen Prozessinstanzen zusätzlich über eine Markierung, die den aktuellen Zustand, in dem sich die Instanz befindet, widerspiegelt. Diese Markierung basiert auf wohl definierten Markierungsregeln [Reichert & Dadam 1998, Reichert 2000b]. Die Markierungsfunktion $M^{WI} = (NS^{WI}, ES^{WI})$ ordnet jeder Aktivität $n \in N^{WI}$ von W_I einen Status $NS^{WI}(n)$ zu; jede Kontrollflusskante $e \in CtrlE^{WI} \cup SyncE^{WI}$ von W_I erhält ebenfalls eine Markierung $ES^{WI}(e)$. Für Datenobjekte gibt es eine Versionierung, was zum einen den kontextsensitiven Zugriff auf verschiedene Versionen von Datenobjekten ermöglicht und zum anderen die Voraussetzung für das Zurücksetzen der Prozessinstanz darstellt. Die folgende formale Spezifikation von Prozessinstanzen entspricht der Definition aus [Rinderle 2004].

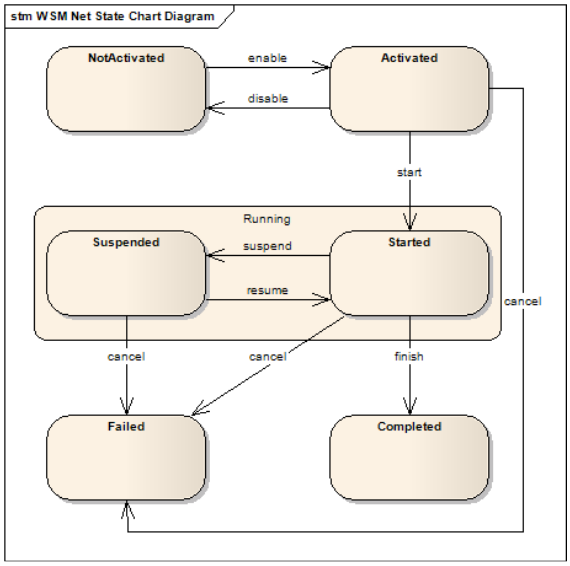
Definition (Prozessinstanz). Eine Prozessinstanz I ist definiert als ein Tupel $I = (W_I, \Delta_I, M^{WI}, Val^{WI}, \Pi^{WI}_I)$, für das gilt:
- $W_I = (N^{WI}, D^{WI}, NT^{WI}, ...)$ (Originäre Prozessdefinition als WSM Net)
- Δ_I (Geordnete, endliche Menge instanz-spezifischer Änderungen $op^I_1, ..., op^I_n$ an der originären Prozessdefinition)
- $M^{WI} = (NS^{WI}, ES^{WI})$ (Markierungsfunktion für Aktivitäten und Kontrollflusskanten)
- $Val^{WI}: D^{WI} \rightarrow \{Undefined\}$ (Datenwertfunktion für instanz-spezifische Datenobjekte)
- Π^{WI}_I (Ausführungshistorie für Prozessinstanzen als geordnete, endliche Menge über Ereignisse $e^I_1, ..., e^I_m$)

Die Elemente haben die folgende Semantik:
- Bei W_I handelt es sich um eine Kopie der originären Prozessdefinition, die aus der Transformation des SPF-Graphen hervorgeht und von der die Instanz abgeleitet wird.

- Δ_I entspricht der Menge instanz-spezifischer Änderungsoperationen, die an W_I durchgeführt wurden. Gilt $\Delta_I = \varnothing$, erfolgten keine Modifikationen an der Instanz.
- $M^{WI} = (NS^{WI}, ES^{WI})$ ist die Markierungsfunktion, die jeder Aktivität und jeder Kontrollflusskante der instanz-spezifischen Prozessdefinition W_I einen Status zuordnet. NS^{WI} und ES^{WI} sind dabei folgendermaßen definiert:
  $NS^{WI}: N_I \mapsto \{NotActivated, Activated, Running, Completed, Failed\}$
  $ES^{WI}: CtrlE_I \cup SyncE_I \mapsto \{NotSignaled, TrueSignaled, FalseSignaled\}$
- $Val^{WI}: D^{WI} \mapsto \{Undefined\}$ entspricht der Datenwertfunktion für instanz-spezifische Datenobjekte und liefert den aktuellen Wert des angefragten Datenobjektes oder *Undefined* zurück.
- $\Pi^{WI}_I = <e^I_1, ..., e^I_m>$ entspricht der Ausführungshistorie für I. $e^I_1, ..., e^I_m$ markieren die Start- und Endereignisse im Hinblick auf die Ausführung einer Prozessaktivität. Für jedes Ereignis werden auch die gelesenen und geschriebenen Werte der Datenobjekte protokolliert.

Die folgende Abbildung zeigt das für ADEPT2 spezifizierte, vereinfachte Zustandsübergangsdiagramm für Instanzen von Prozessfragmenten in WSM Nets.

Abbildung 103 Vereinfachtes Zustandsübergangsdiagramm für Instanzen von Prozessfragmenten in ADEPT2



Nach dem Erzeugen einer Prozessinstanz basierend auf einem WSM Net, wird eine logische Kopie der ursprünglichen Prozessdefinition erzeugt. Außerdem werden alle Knoten und Kanten dieser Kopie mit einer Anfangsmarkierung versehen. Initial befindet sich jede Instanz einer Aktivität im Zustand *NotActivated*; allen Kontrollflusskanten ist die Markierung *NotSignaled* zugewiesen. Es gilt also:

$\forall n \in N^{WI}: NS^{WI}(n) := NotActivated$
$\forall e \in CtrlE^{WI} \cup SyncE^{WI}: ES^{WI}(e) := NotSignaled$

In diesem Zustand ist es nicht möglich, die mit dem Fragment verknüpfte Implementierung auszuführen. Startet ein berechtigter Akteur die Prozessinstanz, so wechselt die Startaktivität automatisch in den Zustand *Completed*; die von ihr ausgehende Kontrollkante erhält die Markierung *TrueSignaled*:

$n \in N^{WI}: NT^{WI}(n) = StartFlow \Rightarrow NS^{WI}(n) := Completed$
$e \in CtrlE^{WI} \cup SyncE^{WI}: NT^{WI}(source(e)) = StartFlow \Rightarrow ES^{WI}(e) := TrueSignaled$

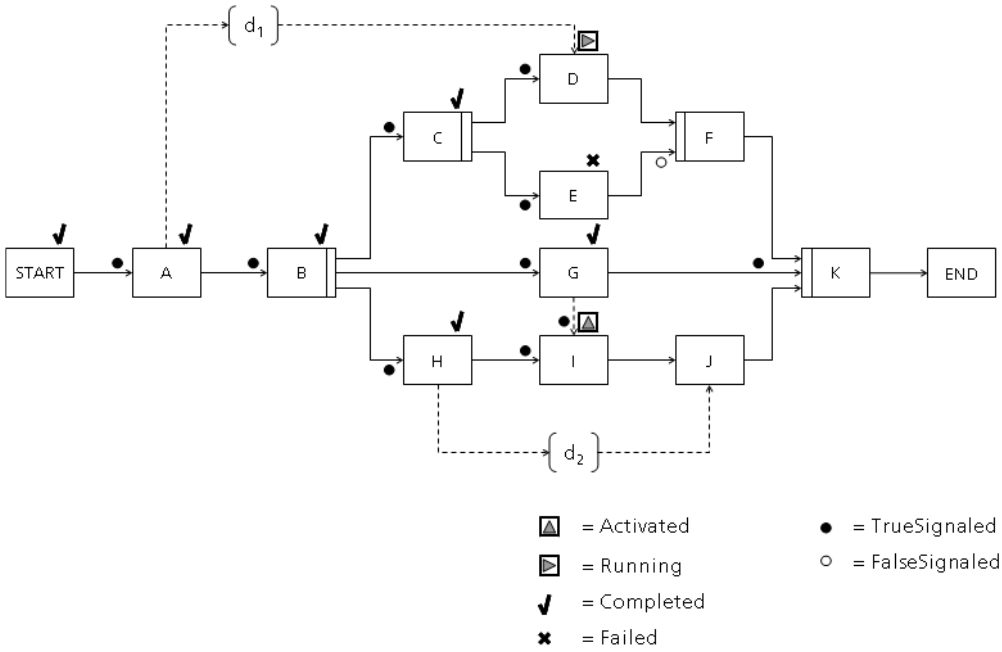
Ändert sich eine Markierung innerhalb der Prozessinstanz, hat dies die Neubewertung der bisherigen Markierungen auf Basis der Markierungsregeln zur Folge. Sämtliche Markierungsregeln sind in [Reichert 2000b][44] detailliert beschrieben. An dieser Stelle wird daher nur ein Beispiel für eine Markierungsregel vorgestellt:

> *Erhält die eingehende Kontrollflusskante e = (m, n) einer Aktivität $n \in N^{WI}$ mit $m \in pred(n)$ und $NS^{WI}(n) = NotActivated$ die Markierung $ES^{WI}(e) := TrueSignaled$, dann wird die Markierung der Aktivität auf $NS^{WI}(n) := Activated$ gesetzt; d. h. die Aktivität ist nun ausführbar.*

Mit den Markierungsregeln werden also gleichzeitig die Bedingungen spezifiziert, unter denen sich der Zustand der Instanz einer Aktivität ändert. Sofern alle Voraussetzungen erfüllt sind, erhält die Instanz den Status *Activated* und darf nun von einem Akteur ausgeführt werden. Wird die Aktivität von einem Fachanwender zur Bearbeitung selektiert, wechselt sie in den Zustand *Started*; wird ihre Ausführung unterbrochen, nimmt sie den Status *Suspended* an. War die Ausführung schließlich erfolgreich, wechselt die Instanz in den Zustand *Completed*. Ist die Durchführung einer Aktivität hingegen nicht möglich, z. B. weil sich der Gesundheitszustand des Patienten radikal geändert hat, erhält sie den Status *Failed*. Ändert sich der Zustand einer Instanz nach *Completed* oder *Failed*, so wird auch der Zustand der nachfolgenden Kontrollflusskanten angepasst. Wurde eine Aktivität erfolgreich abgeschlossen, erhalten die nachfolgenden Kanten den Zustand *TrueSignaled*; alle Kanten, die von einer Aktivität im Zustand *Failed* ausgehen, nehmen den Status *FalseSignaled* an. Die folgende Graphik demonstriert ausgehend von dem WSM Net in Abbildung 77 beispielhaft den Zustand einer Prozessinstanz gemäß ihrer aktuellen Knoten- und Kantenmarkierungen.

[44] Siehe Reichert 2000, S. 93-98

Abbildung 104 Beispiel für eine Prozessinstanz auf Basis eines WSM Nets



Sei $I = (W_I, \Delta_I, M^{WI}, ...)$ eine Prozessinstanz auf Basis eines korrekten WSM Net $W = (N^W, D^W, NT^W, ...)$. Weiterhin seien $Start \in N^{WI}$ mit $NT^{WI}(Start) = StartFlow$ der Startknoten und $M_0^{WI} = (NS_0^{WI}, ES_0^{WI})$ die Anfangsmarkierung mit $NS_0^{WI}(Start) = Activated$, $\forall n \in N^{WI} \setminus \{Start\}: NS_0^{WI}(n) = NotActivated$ und $\forall e \in CtrlE^{WI} \cup SyncE^{WI}: ES_0^{WI}(e) = NotSignaled$. Eine beliebige Markierung M^{WI} der Prozessinstanz ist gültig, wenn sie ausgehend von der Startmarkierung nach der Anwendung einer Menge von Markierungsregeln erreicht werden kann:

$$\exists \langle e_0^I, ..., e_m^I \rangle (e_i^I \in Start(n), End(n)\}, n \in N_I) \text{ mit } M_0^{WI}[e_0^I \rangle, M_1^{WI}, ..., [e_m^I \rangle M_k^{WI} =: M^{WI}$$

Folglich muss eine geordnete Sequenz von Ereignissen existieren, die jeweils den Beginn *Start(n)* und das Ende *End(n)* der Ausführung eines Prozessfragments markieren, deren Erscheinen dazu führt, dass eine Historie von Markierungen hervorgeht, die letztlich in dem Entstehen von M^{WI} mündet; der Ausdruck $M[e \rangle, M'$ bedeutet, dass die Markierung M' nach dem Eintritt des Ereignisses *e* bei Markierung *M* entsteht. Damit eine Markierung M^{WI} korrekt ist, muss zusätzlich die folgende Bedingung erfüllt sein:

$$\forall n \in N^{WI}: NS^{WI}(n) \in \{Activated, Running, Completed, Failed\} \Rightarrow \forall u \in pred^*(n): NS^{WI}(u) \in \{Completed\}$$

Das bedeutet, die Instanz eines Prozessfragments kann nur dann einen anderen Status als *NotActivated* annehmen, wenn die Bearbeitung ihrer Vorgängeraktivitäten erfolgreich abgeschlossen wurde.

4.6.2.2 Bedingungen für die Rekonfiguration

Ausgehend von den in Kapitel 4.6.2.1 spezifizierten Prozessinstanzen von ADEPT2 WSM Nets ist es nun möglich, die Bedingungen anzugeben, die für die Rekonfiguration zur Laufzeit und die Transformation von Prozessinstanzen erfüllt sein müssen. Anders als bei der Rekonfi-

guration zur Modellierungszeit, stehen Fachanwender bei der Rekonfiguration zur Laufzeit meist unter Zeitdruck, da schnell eine Entscheidung darüber getroffen werden muss, ob der bisher geplante Prozess noch das Optimum für einen Behandlungsfall darstellt oder ob Modifikationen erforderlich sind. Durch gezielte Modifikation soll der Prozess an die aktuellen Erfordernisse der Patientenbehandlung angepasst werden, sie soll aber nicht die Prozessplanung und die laufenden Arbeiten unnötig erschweren. Aus diesem Grund müssen Fachanwender vom System darauf hingewiesen werden, wenn die von ihnen geplanten Rekonfigurationen unter Umständen unerwünschte Auswirkungen auf unterschiedliche Teile der Prozessinstanzen haben.

Generell gilt, dass Knoten immer durch Rekonfiguration in den SPF-Graphen eingefügt werden dürfen, sofern der SPF-Typ-Graph dies erlaubt. Auf Grund des fortgeschrittenen Ausführungsstatus einer Prozessinstanz kann es aber vorkommen, dass ein Prozessfragment, das im SPF-Typ-Graphen im Fall einer Abweichung eigentlich während der Aufnahmephase vorgesehen ist, tatsächlich erst am Operationstag durchgeführt wird. An der kontextuellen Zuordnung im SPF-Graphen ändert sich dadurch nichts; lediglich der Zeitpunkt der Ausführung des Fragments in der Prozessdefinition verschiebt sich. Diese Form der Flexibilität ist bei den komplexen klinischen Prozessen, in denen der Zeitpunkt der Durchführung einer Aktivität stark variieren kann, unbedingt notwendig. Aus diesem Grund wäre es zu kurzsichtig, wenn ein Knoten nur solange in den SPF-Graphen eingefügt werden könnte, bis alle Prozessfragmente, die seinem übergeordneten Kontextknoten zugeordnet sind, abgearbeitet wurden. Die in Kapitel 4.6.1.1 definierten Bedingungen für die Rekonfiguration zur Modellierungszeit werden nun im Hinblick auf die Rekonfiguration zur Laufzeit erweitert.

Hinzufügen von Klonknoten

Bei ADEPT2 gilt eine Prozessinstanz als abgeschlossen, sobald der Endknoten im Zustand *Completed* ist; in diesen Zustand geht der Endknoten sofort nach Aktivierung über. Sobald die Instanz abgeschlossen ist, können keine Änderungen mehr an ihr vorgenommen werden. Daher gilt als grundlegende Bedingung für das Einfügen von Klonknoten, dass der Endknoten sich im Zustand *NotActivated* befindet. Sei $I = (W_I, \Delta_I, M^{WI}, Val^{WI}, \Pi^{WI}_I)$ die zu transformierende WSM Net Instanz, dann gilt:

$$\forall n \in N^{WI}: NT^{WI}(n) = EndFlow \Rightarrow NS^{WI}(n) = NotActivated$$

Ansonsten gelten beim Hinzufügen von Klonknoten in den SPF-Graphen für Prozessinstanzen exakt dieselben Regeln wie für Prozessdefinitionen. D. h. ein Klonknoten darf auch dann hinzugefügt werden, wenn er nicht mehr sequentiell hinter den bereits existierenden Knoten in die Prozessinstanz eingefügt werden kann. Wie bei der Transformation von Prozessinstanzen in Kapitel 4.6.2.4 gezeigt wird, werden alle bei der Rekonfiguration neu entstandenen Knoten immer an der nächstmöglichen Stelle in die Instanz eingefügt.

Bei der im Anschluss durchzuführenden Konfiguration des Klonknoten gilt zu beachten, dass das Einhalten von Constraint-Relationen nicht automatisch erzwungen wird, sondern die Konfiguration genau wie die Rekonfiguration nicht möglich ist, wenn dadurch Constraint-Bedingungen verletzt werden; wie dies überprüft werden kann, wird im nächsten Abschnitt dargestellt.

Hinzufügen von neuen Knoten

Auch für das Einfügen von neuen Knoten in den SPF-Graphen gilt die Bedingung, dass sich der Endknoten im Zustand *NotActivated* befinden muss. Während das Hinzufügen von Klon-

knoten jedoch unkritisch ist, kann das Hinzufügen komplett neuer Knoten zu Problemen führen, wenn sie über Ausschluss-Constraints mit anderen Knoten verbunden sind, die bereits im SPF-Graphen existieren. Wurde die Ausführung solcher Konflikt-auslösenden Knoten gestartet oder ist sie bereits abgeschlossen, ist der Einfügevorgang generell nicht möglich. Selbst wenn das Löschen des Knotens im Hinblick auf seinen Ausführungsstatus zulässig ist, darf dies nicht wie bei der Konfiguration oder der Rekonfiguration zur Modellierungszeit automatisch erfolgen, sondern muss durch die Initiative des Fachanwenders eingeleitet werden. Daher gilt für die Rekonfiguration zur Laufzeit, dass ein Knoten dem SPF-Graphen nicht hinzugefügt werden darf, so lange der Graph noch einen Knoten enthält, der in einer Ausschlussbeziehung mit dem einzufügenden Knoten steht; dieser Konflikt-auslösende Knoten muss also zuerst manuell aus dem SPF-Graphen entfernt werden. Sei $STG = (V^{STG}, E^{STG}, ...)$ ein SPF-Typ-Graph und $SG = (V^{SG}, E^{SG}, ...)$ ein davon abgeleiteter SPF-Graph; die Menge V^{STG}_{Add} sei wie in Kapitel 4.6.1.1 definiert und $v^{STG}_{Add} \in V^{STG}_{Add}$ der Kandidat für das Einfügen in den SPF-Graphen. Dann kann die Menge möglicher Konflikt-auslösender Knoten wie folgt festgelegt werden:

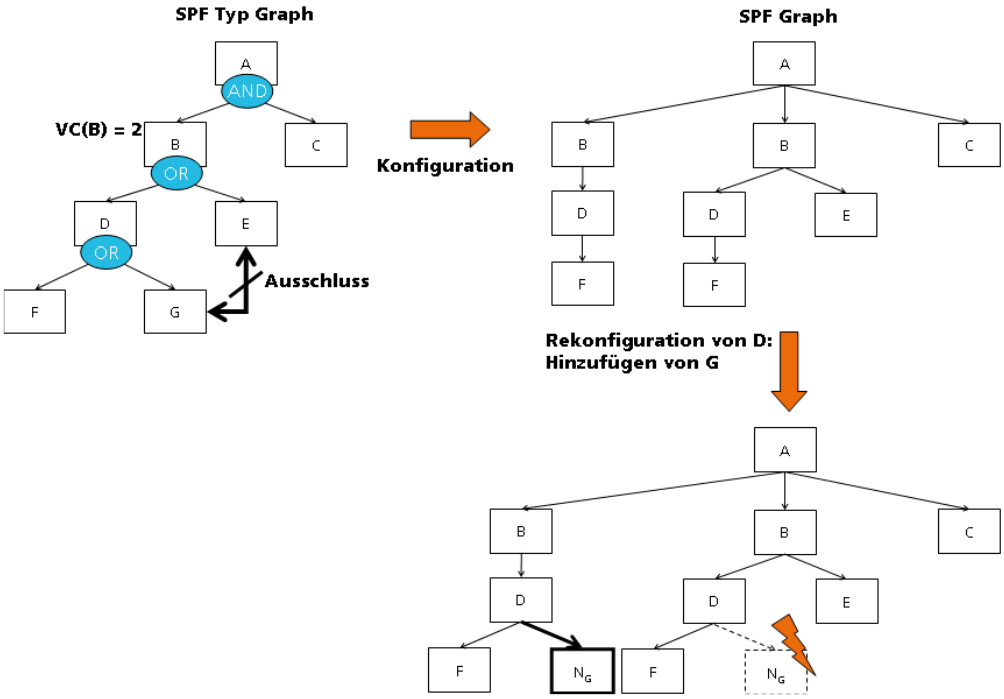
$$V^{STG}_{Exc} = \{v^{STG} \in V^{STG} | (v^{STG}, v^{STG}_{Add}, EXCLUSION) \in C^{STG} \vee (v^{STG}_{Add}, v^{STG}, EXCLUSION) \in C^{STG}\}$$

Echtes Konfliktpotential besteht jedoch nur dann, wenn es sich bei dem Rekonfigurationsknoten entweder um den kleinsten gemeinsamen Kontextknoten von v^{STG}_{Add} und einem Knoten der Menge V^{STG}_{Exc} im SPF-Graphen handelt oder um den Nachfolger dieses kleinsten gemeinsamen Knoten. Als Bedingung für das Einfügen eines neuen Knotens muss also zusätzlich gelten:

$$\nexists\ v^{STG}_{Exc} \in V^{STG}_{Exc}\ \exists\ v^{SG}, min^{SG} \in V^{SG}: ((g_v(v^{SG}) = v^{STG}_{Exc} \vee \exists\ (v^{STG}_{Exc}, L, R) \in PR^{GG}: L = (v^{SG}, \varnothing, \varnothing, v^{SG}, v^{SG}, A^{SG}, VA^{SG}, VID^{SG})) \wedge g_v(min^{SG}) = minNode(v^{STG}_{Add}, v^{STG}_{Exc})) \wedge (min^{SG} = v_{Reconfig} \vee min^{SG} \rightarrow^* v_{Reconfig}) \wedge min^{SG} \rightarrow^* v^{SG}$$

Die nachfolgende Abbildung illustriert die Auswirkungen von Ausschluss-Constraints auf das Einfügen von Knoten im Kontext der Rekonfiguration zur Laufzeit beispielhaft.

Abbildung 105 Auswirkung des Ausschluss-Constraints auf das Einfügen von Knoten während der Rekonfiguration zur Laufzeit



Im SPF-Typ-Graphen ist zwischen Knoten *E* und *G* ein Ausschluss-Constraint definiert. Der SPF-Graph enthält zwei Klone von *B*. Nur der zweite Klon umfasst auch den Constraint-relevanten Knoten *E*. Alle Klone von *D* sollen nun so rekonfiguriert werden, dass sie den Knoten *G* enthalten. Bei dem ersten Klon ist dies problemlos möglich, da es unterhalb des kleinsten gemeinsamen Kontextknoten von *E* und *G*, nämlich *B*, keinen Knoten *E* gibt. Beim zweiten Klon ist die Rekonfigurationsoperation hingegen nicht erlaubt.

Das Einfügen eines neuen Knotens ist auch dann nicht möglich, wenn dieser Knoten über einen Anforderungs-Constraint mit weiteren Knoten verbunden ist, die bisher nicht in der Prozessdefinition vorkommen. Würde der erforderliche Knoten automatisch in den SPF-Graphen eingefügt, könnte dies wiederum zu hochgradig unerwünschten Konsequenzen führen. Ist der erforderliche Knoten z. B. über einen Ausschluss-Constraint mit einem dritten Knoten verbunden, dann würden alle mit ihm assoziierten Prozessfragmente aus der Prozessdefinition entfernt werden. Um solche kaskadierenden Änderungen zur Laufzeit zu vermeiden, darf ein Knoten durch Rekonfiguration nicht in den SPF-Graphen eingefügt werden, solange er andere Knoten erfordert, die nicht im Graphen enthalten sind; d. h. der Fachanwender muss manuell und kontrolliert zuerst alle Änderungen vornehmen, die dazu führen, dass der ausgewählte Knoten dem SPF-Graphen hinzugefügt werden darf. Sein nun $v^{STG}_{Add} \in V^{STG}$ wiederum der neu einzufügende Knoten. Die Menge der Knoten, die aufgrund eines nicht erfüllten Anforderungs-Constraints problematisch sein können, lässt sich folgendermaßen definieren:

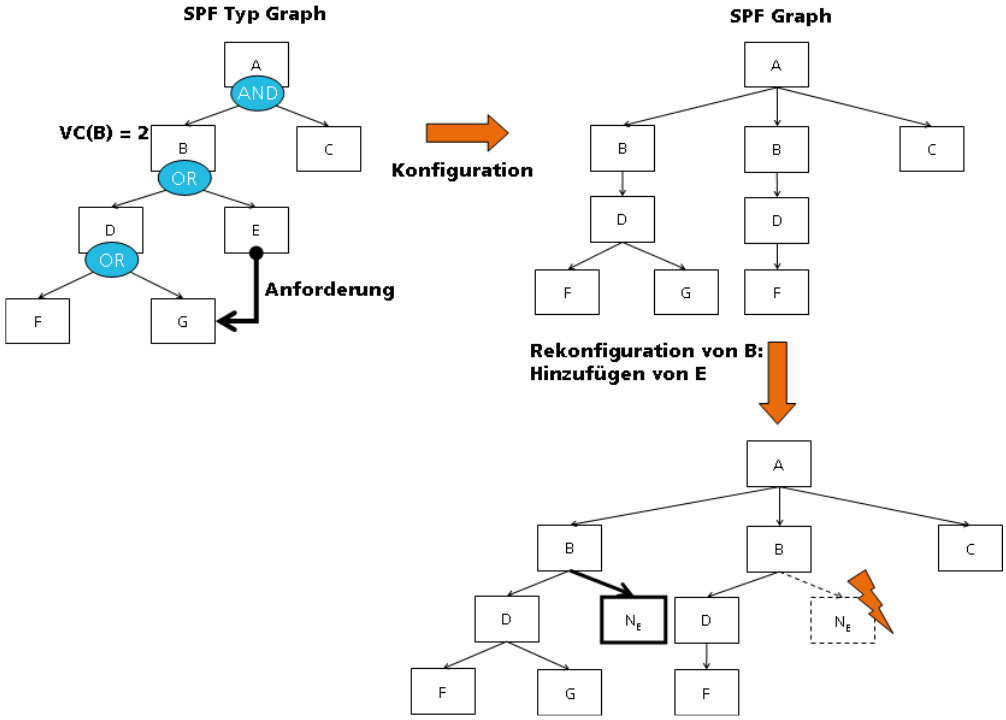
$$V^{STG}_{Req} = \{v^{STG} \in V^{STG} | (v^{STG}_{Add}, v^{STG}, REQUIREMENT) \in C^{STG}\}$$

Analog zur Berücksichtigung von Ausschluss-Constraints ist das Einfügen von v^{STG}_{Add} nur dann möglich, wenn es sich bei dem Rekonfigurationsknoten weder um den kleinsten gemeinsamen Kontextknoten von v^{STG}_{Add} und einem Knoten der Menge V^{STG}_{Req} im SPF-Graphen handelt, noch um dessen Nachfolger:

$$\forall v^{STG}_{Req} \in V^{STG}_{Req}\ \exists v^{SG}, min^{SG} \in V^{SG}: ((g_v(v^{SG}) = v^{STG}_{Req} \vee \exists (v^{STG}_{Req}, L, R) \in PR^{GG}: L = (v^{SG}, \varnothing, \varnothing, v^{SG}, v^{SG}, A^{SG}, VA^{SG}, VID^{SG})) \wedge g_v(min^{SG}) = minNode(v^{STG}_{Add}, v^{STG}_{Req})) \wedge (min^{SG} = v_{Reconfig} \vee min^{SG} \rightarrow^* v_{Reconfig})) \wedge min^{SG} \rightarrow^* v^{SG}$$

Die nächste Abbildung verdeutlicht diesen Zusammenhang beispielhaft.

Abbildung 106 Auswirkung des Anforderungs-Constraints auf das Einfügen von Knoten während der Rekonfiguration



Im SPF-Typ-Graphen ist nun zwischen Knoten *E* und *G* ein Anforderungs-Constraint definiert. Im Zuge der Rekonfiguration soll nun unterhalb von Knoten *B* jeweils ein neuer Knoten *E* hinzugefügt werden. Beim ersten Klonknoten von *B* ist dies problemlos möglich, da der Subgraph auch den Knoten *B* umfasst. Im Hinblick auf den zweiten Klon darf *E* hingegen nicht eingefügt werden, da *G* nicht unterhalb des Rekonfigurationsknoten *B* vorhanden ist.

Bei der Konfiguration des neu hinzugefügten Knoten gelten im Hinblick auf Constraint-Relationen dieselben Regeln wie bei der Rekonfiguration; d. h. die Produktionsregel einer Graphgrammatik darf nicht angewendet werden, wenn dadurch Ausschluss-Constraints verletzt werden oder Anforderungs-Constraints nicht erfüllt sind. Zur Überprüfung können dieselben, in diesem Abschnitt aufgestellten Bedingungen genutzt werden.

Entfernen von Klonknoten

Beim Entfernen von Klonknoten gilt neben den in Kapitel 4.6.1.1 spezifizierten Bedingungen zusätzlich die Einschränkung, dass mit der Bearbeitung des mit dem Knoten assoziierten Prozessfragments noch nicht begonnen wurde bzw. dass die Ausführung bereits vollendet ist. Diese Bedingung kann nur spezifisch im Hinblick auf die Ausführungssemantik des WfMS formuliert werden, für das die Transformation zuvor stattfand. In Kapitel 4.6.2.1 wurden mögli-

che Zustände und Zustandsübergänge von Aktivitäten in Instanzen von WSM Nets dargestellt. Seien *STG*, *SG* und *I* wie oben gegeben. Sei V^{SG}_{Clon} analog zu der in Kapitel 4.6.1.1 spezifizierten Menge von Klonknoten definiert, die potentiell aus dem SPF-Graphen gelöscht werden können. Ein Knoten $v^{SG} \in V^{SG}_{Clon}$ darf zur Laufzeit nur dann aus dem SPF-Graphen entfernt werden, wenn die folgende Bedingung erfüllt ist:

$$\forall u^{SG} \in V^{SG}_{T}: g_v(u^{SG}) \in V^{STG}_{SPF} \wedge (u^{SG} = v^{SG} \vee v^{SG} \rightarrow^* u^{SG}) \wedge (\exists n \in N^{WI}: n = Name^p \cup VID^{SG}(u^{SG}) \text{ mit } p = VP^{STG}(g_v(u^{SG}))) \Rightarrow NS^{WI}(n) \notin \{Running, Completed, Failed\}$$

Diese Bedingung besagt, dass der Knoten nur dann gelöscht werden darf, wenn alle mit ihm bzw. seinen Nachfolgern verknüpften Prozessfragmente nicht im Zustand *Running*, *Completed* oder *Failed* sind. Da nicht-terminale Knoten nicht in Platzhalteraktivitäten in WSM Nets transformiert werden, sind sie in dieser Definition nicht berücksichtigt. Gibt es Platzhalteraktivitäten für Nichtterminalknoten, so muss deren Ausführungsstatus ebenfalls berücksichtigt werden.

Entfernen von einzelnen Knoten

Die Menge der potentiell zu entfernenden einzelnen Knoten V^{SG}_{Del} sei analog zu Kapitel 4.6.1.1 definiert. Wie bei Klonknoten muss für alle Elemente $v^{SG}_{Del} \in V^{SG}_{Del}$ dieser Menge und den ihnen zugeordneten Nachfolgerknoten folgende Bedingung erfüllt sein, damit sie durch Rekonfiguration zur Laufzeit gelöscht werden können:

$$\forall u^{SG} \in V^{SG}_{T}: g_v(u^{SG}) \in V^{STG}_{SPF} \wedge (u^{SG} = v^{SG}_{Del} \vee v^{SG}_{Del} \rightarrow^* u^{SG}) \wedge (\exists n \in N^{WI}: n = Name^p \cup VID^{SG}(u^{SG}) \text{ mit } p = VP^{STG}(g_v(u^{SG}))) \Rightarrow NS^{WI}(n) \notin \{Running, Completed, Failed\}$$

Wie in Kapitel 4.6.1.1 dargestellt wurde, können logische Operatoren das Entfernen von Rekonfigurationsknoten beeinflussen. Bei der Rekonfiguration zur Laufzeit müssen darüber hinaus Anforderungs-Constraints berücksichtigt werden. So darf ein Knoten und der mit ihm assoziierte Teilgraph nicht aus dem SPF-Graphen entfernt werden, wenn er von einem anderen Knoten benötigt wird, der nicht gelöscht wird. Das Entfernen des abhängigen Knotens stellt hingegen grundsätzlich kein Problem dar. Die Menge der Knoten, die der Löschung des von v^{SG}_{Del} aus dem SPF-Graphen verhindern könnten, lässt sich folgendermaßen definieren:

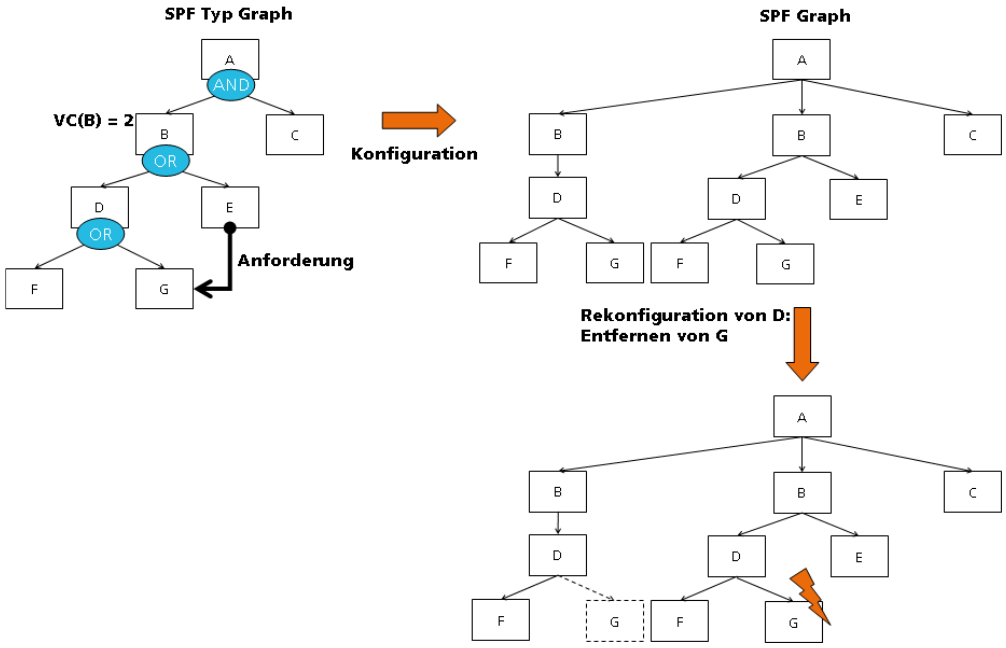
$$V^{STG}_{Req} = \{u^{STG} \in V^{STG} | (u^{STG}, g_v(v^{SG}_{Del}), REQUIREMENT) \in C^{STG}\}$$

Analog zur Berücksichtigung von Ausschluss-Constraints ist das Löschen von v^{SG}_{Del} nur dann möglich, wenn es sich bei dem Rekonfigurationsknoten weder um den kleinsten gemeinsamen Kontextknoten von v^{SG}_{Del} und einem Knoten der Menge V^{STG}_{Req} im SPF-Graphen handelt, noch um dessen Nachfolger:

$$\nexists v^{STG}_{Req} \in V^{STG}_{Req} \exists v^{SG}, min^{SG} \in V^{SG}: ((g_v(v^{SG}) = v^{STG}_{Req} \vee \exists (v^{STG}_{Req}, L, R) \in PR^{GG}: L = (v^{SG}, \varnothing, \varnothing, v^{SG}, v^{SG}, A^{SG}, VA^{SG}, VID^{SG})) \wedge g_v(min^{SG}) = minNode(v^{STG}_{Del}, v^{STG}_{Req})) \wedge (min^{SG} = v_{Reconfig} \vee min^{SG} \rightarrow^* v_{Reconfig}) \wedge min^{SG} \rightarrow^* v^{SG}$$

Die nachfolgende Abbildung verdeutlicht die Auswirkung des Anforderungs-Constraint bei der Rekonfiguration an einem Beispiel.

Abbildung 107 Auswirkung des Anforderungs-Constraints beim Löschen eines Knotens während der Rekonfiguration



Der SPF-Typ-Graph definiert einen Anforderungs-Constraint von *E* nach *G*. Der SPF-Graph enthält zwei Knoten von Typ *B*, wobei nur der zweite auch den abhängigen Knoten *E* umfasst. Sollen nun alle Klone von *D* rekonfiguriert werden, indem Nachfolger *G* entfernt wird, so ist das nur bei dem ersten Klon möglich. Bei dem zweiten Klon kommt unterhalb des kleinsten gemeinsamen Kontextknotens von *E* und *G*, nämlich *B*, ein Knoten *E* vor, so dass der Constraint durch das Entfernen von *G* an dieser Stelle verletzt werden würde; *G* darf hier also nicht aus dem SPF-Graphen gelöscht werden.

4.6.2.3 Strukturerhaltende und nicht-strukturerhaltende Rekonfiguration

Die in Kapitel 4.6.1.2 spezifizierten Algorithmen müssen nur geringfügig modifiziert werden, um auch die Rekonfiguration zur Laufzeit zu unterstützen. So ist die nicht-strukturerhaltende Rekonfiguration nur für solche Knoten möglich, bei denen sich keines der mit ihnen oder ihren Nachfolgern assoziierten Prozessfragmente im Zustand *Running*, *Completed* oder *Failed* befindet. Der Funktion r*econfigureNodeRunTime* werden neben dem SPF-Graphen *SG* und dem Rekonfigurationsknoten $v_{Reconfig}$ auch der SPF-Typ-Graphen *STG* und die entsprechende Prozessinstanz $I = (W_I, \Delta_I, M^{WI}, Val^{WI}, \Pi^{WI}_I)$ als Parameter übergeben.

Formel 21 Algorithmus zur nicht-strukturerhaltenden Rekonfiguration einer Knotenmenge zur Laufzeit

```
boolean reconfigureNodeRunTime(STG, SG, I, v_Reconfig) {
      IF (v_Reconfig ∈ V^SG_T ∧ g_v(v_Reconfig) ∉ V^STG_SPF)
         SPFNodes := {v^SG ∈ V^SG_T | v_Reconfig →* v^SG ∧ g_v(v^SG) ∈ V^STG_SPF};
         N := {n ∈ N^WI | ∃v^SG ∈ SPFNodes ∃p = VP^STG(g_v(v^SG)): n = Name^p ∪
VID^SG(v^SG) ∧ NS^WI(n) ∈ {Running, Completed, Failed}};
         IF (N = ∅ )
            //Nicht-strukturelle Rekonfiguration des Knoten
            sourceNode^SG := {v ∈ V^SG | v → v_Reconfig};
```

```
            succNodes^SG := {v ∈ V^SG|v_Reconfig →* v};
            E^SG := E^SG - {(x^SG, y^SG) ∈ E^SG|x ∈ (sourceNode^SG ∪ v_Reconfig ∪
succNodes^SG) ∧ y^SG ∈ (v_Reconfig ∪ succNodes^SG)};
            V^SG := V^SG - (v_Reconfig ∪ succNodes^SG);
            V^SG := V^SG ∪ N_vReconfig;
            E^SG := E^SG ∪ (sourceNode^SG, N_vReconfig);
         END IF
      END IF
}
```

Beim Hinzufügen eines neuen Knotens bzw. eines Klonknotens mit Strukturerhalt müssen neben dem Typ des logischen Operators und den Knotenkardinalitäten auch die geltenden Constraint-Relationen berücksichtigt werden. Die Funktion erhält als zusätzliche Parameter die zu transformierende Prozessinstanz.

Formel 22 Algorithmus für das strukturerhaltende Einfügen nicht-terminaler Knoten in SPF-Graphen zur Laufzeit

```
addNodeRunTime(STG, SG, I, v_Reconfig, v^STG_Add) {
      n := {n ∈ N^WI|NT^WI(n) = EndFlow}
      IF(NS^WI(n) = NotActivated ∧ v_Reconfig ∈ V^SG)
         IF (v^STG_Add ∈ target(g_v(v_Reconfig)))
            maxCardinality := VC^STG(v^STG_Add);
            V_Clon := {v^SG ∈ target(v_Reconfig)|g_v(v^SG) = v^STG_Add ∨ (∃(v^STG_Add, L, R) ∈
PR^GG: L = (v^SG, ∅ , ∅ , v^SG, v^SG, A^SG, VA^SG, VID^SG))};
            IF (|V_Clon| > 0 ∧ maxCardinality - |V_Clon| > 0)
               //Einfügen des Nichtterminalknotens als Klonknoten
               V^SG := V^SG ∪ N^1_vSTGAdd;
               E^SG := E^SG ∪ (v_Reconfig, N^1_vSTGAdd);
            END IF
            ELSE IF (|V_Clon| = 0 ∧ VO^STG(g_v(v_Reconfig)) ∈ {OR, OPT}) {
               //Menge möglicher konflikt-auslösender Knoten
               V^STG_Exc := {v^STG ∈ V^STG|(v^STG, v^STG_Add, EXCLUSION) ∈ C^STG ∨ (v^STG_Add, v^STG,
EXCLUSION) ∈ C^STG};
               //Menge möglicher erforderlicher Knoten
               V^STG_Req := {v^STG ∈ V^STG|(v^STG_Add, v^STG, REQUIREMENT) ∈ C^STG};
               IF (∄ v^STG_Exc ∈ V^STG_Exc ∃v^SG, min^SG ∈ V^SG: ((g_v(v^SG) = v^STG_Exc ∨ ∃(v^STG_Exc,
L, R) ∈ PR^GG: L = (v^SG, ∅ , ∅ , v^SG, v^SG, A^SG, VA^SG, VID^SG)) ∧ g_v(min^SG) =
minNode(v^STG_Add, v^STG_Exc)) ∧ (min^SG = v_Reconfig ∨ min^SG →* v_Reconfig) ∧ min^SG →* v^SG)
                  IF (∀v^STG_Req ∈ V^STG_Req ∃v^SG, min^SG ∈ V^SG: ((g_v(v^SG) = v^STG_Req ∨ ∃
(v^STG_Req, L, R) ∈ PR^GG: L = (v^SG, ∅ , ∅ , v^SG, v^SG, A^SG, VA^SG, VID^SG)) ∧ g_v(min^SG) =
minNode(v^STG_Add, v^STG_Req)) ∧ (min^SG = v_Reconfig ∨ min^SG →* v_Reconfig) ∧ min^SG →* v^SG)
                     //Einfügen des Nichtterminalknotens als neuen Knoten
                     V^SG := V^SG ∪ N_vSTGAdd;
                     E^SG := E^SG ∪ (v_Reconfig, N_vSTGAdd);
                  END IF
               END IF
            END ELSE IF
         END IF
      END IF
}
```

Während sich das Hinzufügen eines Klonknoten in den SPF-Graphen zur Laufzeit nicht von der Vorgehensweise zur Modellierungszeit unterscheidet, ist das Einfügen eines neuen Knoten an zusätzliche Bedingungen geknüpft. Nach der Überprüfung des Ausführungsstatus der Instanz wird die Menge aller Knoten im SPF-Typ-Graphen ermittelt, die möglicherweise mit dem neu einzufügenden Knoten in Konflikt stehen könnten. Analog dazu wird auch die Menge aller derjenigen Knoten bestimmt, von denen der neue Knoten abhängt. Anschließend wird geprüft, ob es unterhalb des kleinsten gemeinsamen Kontextknoten keinen Konflikt-auslösen-

den Knoten gibt bzw. ob die erforderlichen Knoten vorhanden sind. Ist dies nicht der Fall, kann der Knoten dem SPF-Graphen hinzugefügt werden.

Als nächstes wird die Funktion zum Entfernen eines Klonknoten bzw. eines einzelnen Knoten zur Laufzeit dargestellt.

Formel 23 Algorithmus für das strukturerhaltende Entfernen von Teilgraphen aus SPF-Graphen zur Laufzeit

```
deleteNodeRunTime(STG, SG, I, v_Reconfig, v^SG_Del) {
      IF (v^SG_Del ∈ target(v_Reconfig))
         //Überprüfen des Ausführungsstatus des Knoten
         SPFNodes := {v^SG ∈ V^SG_T|(v^SG = v^SG_Del ∨ v^SG_Del →* v^SG) ∧ g_v(v^SG) ∈ V^STG_SPF};
         N := {n ∈ N^WI|∃v^SG ∈ SPFNodes ∃p = VP^STG(g_v(v^SG)): n = Name^p ∪
VID^SG(v^SG) ∧ NS^WI(n) ∈ {Running, Completed, Failed};
         IF (N = ∅ )
            V_Clon := {v^SG ∈ target(v_Reconfig)|v^SG ≠ v^SG_Del ∧ (g_v(v^SG) = g_v(v^SG_Del) ∨ (∃
v^STG ∈ V^STG: (v^STG = g_v(v^SG_Del) ∧ ∃(v^STG, L, R) ∈ PR^GG: L = (v^SG, ∅ , ∅ , v^SG, v^SG, VA,
VID)) ∨ (v^STG = g_v(v^SG) ∧ ∃(v^STG, L, R) ∈ PR^GG: L = (v^SG_Del, ∅ , ∅ , v^SG_Del, v^SG_Del, VA,
VID)) ∨ (∃(v^STG, L, R) ∈ PR^GG: L = (v^SG, ∅ , ∅ , v^SG, v^SG, VA, VID) ∧ ∃(v^STG, L,
R) ∈ PR^GG: L = (v^SG_Del, ∅ , ∅ ,v^SG_Del, v^SG_Del, VA, VID)))};
            IF (|V_Clon| > 0)
               //Strukturerhaltendes Entfernen des Knoten
               V_Del := {v ∈ V^SG| v^SG_Del →* v} ∪v^SG_Del;
               E_Del := {(x, y) ∈ E^SG|x, y ∈ V_Del} ∪ (v_Reconfig, v^SG_Del);
               E^SG := E^SG - E_Del;
               V^SG := V^SG - V_Del;
            END IF
            ELSE IF (|V_Clon| = 0 ∧ VO^STG(g_v(v_Reconfig)) ≠ AND)
               IF (VO^STG(g_v(v_Reconfig)) = OPT ∨ (VO^STG(g_v(v_Reconfig)) = OR ∧
|target(v_Reconfig)| > 1))
                  //Feststellen möglicher abhängiger Knoten
                  V^STG_Req := {v^STG ∈ V^STG|(v^STG, g_v(v^STG_Del), REQUIREMENT) ∈ C^STG};
                  IF (∄ v^STG_Req ∈ V^STG_Req ∃v^SG, min^SG ∈ V^SG: ((g_v(v^SG) = v^STG_Req ∨ (∃
(v^STG_Req,  L, R) ∈ PR^GG: L = (v^SG, ∅ , ∅ , v^SG, v^SG, A^SG, VA^SG, VID^SG)) ∧ g_v(min^SG) =
minNode(v^STG_Del, v^STG_Req)) ∧ (min^SG = v_Reconfig ∨ min^SG →* v_Reconfig) ∧ min^SG →* v^SG)
                     //Strukturerhaltendes Entfernen des Knoten
                     V_Del := {v ∈ V^SG| v^SG_Del →* v} ∪v^SG_Del;
                     E_Del := {(x, y) ∈ E^SG|x, y ∈ V_Del} ∪ (v_Reconfig, v^SG_Del);
                     E^SG := E^SG - E_Del;
                     V^SG := V^SG - V_Del;
                  END IF
               END IF
            END ELSE IF
            ELSE
               //Nicht-strukturerhaltende Rekonfiguration bei XOR- oder OR-
Operator ohne weitere Nachfolger
               reconfigureNodeRunTime(STG, SG, I, v_Reconfig);
               return true;
            END ELSE
         END IF
      END IF
}
```

Zuerst wird der Ausführungsstatus aller Prozessfragmente, die entweder direkt mit dem Rekonfigurationsknoten oder indirekt über seine Nachfolger verknüpft sind, überprüft. Gibt es ein Prozessfragment, das sich gerade in Bearbeitung befindet, das bereits abgeschlossen oder fehlgeschlagen ist, darf der Rekonfigurationsknoten und der mit ihm assoziierte Teilgraph nicht aus dem SPF-Graphen entfernt werden. Ansonsten wird ermittelt, ob es sich bei dem Rekonfigurationsknoten um einen Klon handelt; ist dies der Fall, kann er aus dem SPF-Graphen ge-

löscht werden. Gibt es hingegen keine Klone im SPF-Graphen, so wird wie bei der Rekonfiguration zur Modellierungszeit überprüft, ob die Bedingungen im Hinblick auf den logischen Operator des Rekonfigurationsknotens erfüllt sind. Anschließend werden alle Knoten ermittelt, die möglicherweise von dem zu entfernenden Knoten abhängig sein könnten. Nur wenn es keine abhängigen Knoten im SPF-Graphen gibt, ist die Löschung des Knotens möglich. Entspricht die logische Verknüpfung, die dem Rekonfigurationsknoten zugeordnet ist, einem XOR-Operator, oder handelt es sich um einen OR-Operator und verfügt $v_{Reconfig}$ über keine weiteren Nachfolger, so wird die Funktion zur Durchführung der nicht-strukturerhaltenden Rekonfiguration aufgerufen.

4.6.2.4 Transformation von SPF-Graphen nach der Rekonfiguration

Wie bei der Transformation von SPF-Graphen zur Modellierungszeit gibt es auch bei der Transformation zur Laufzeit zwei mögliche Szenarien, die auch in Kombination auftreten können: Es wurden Knoten aus dem SPF-Graphen gelöscht, die nun aus der Prozessinstanz entfernt werden müssen oder es wurden neue Knoten hinzugefügt, die bisher in der Instanz noch nicht existieren. Da beim Löschen von Knoten aus dem SPF-Graphen der Ausführungsstatus der zugehörigen Prozessfragmente in der Prozessinstanz bereits berücksichtigt wird, entspricht der Löschvorgang von Aktivitäten in der Instanz exakt dem Vorgang für Prozessdefinitionen (siehe Kapitel 4.6.1.3). Die gegebenenfalls notwendigen Statusänderungen werden in ADEPT2 durch Auswertung der Markierungsregeln automatisch durchgeführt. Bei dem nachträglichen Entfernen der direkt aufeinanderfolgenden *AndSplit*- und *AndJoin*-Knoten muss allerdings ebenfalls deren Status berücksichtigt werden; sei $I = (W_I, \Delta_I, M^{WI}, Val^{WI}, \Pi^{WI}_I)$ die zu transformierende Prozessinstanz, dann gilt:

$$\forall n \in N^{WI}: NT^{WI}(n) = AndSplit \wedge NT^{WI}(succ(n)) = AndJoin \wedge NS^{WI}(n) \in \{NotActivated, Activated\} \Rightarrow deleteBlock(W_I, n, succ(n))$$

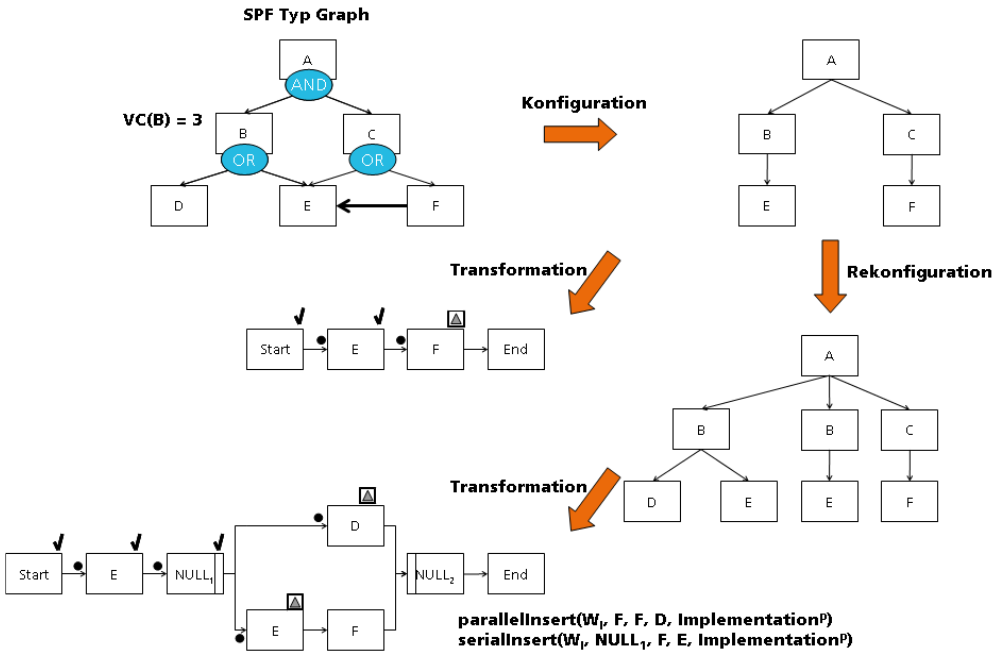
Die neu einzufügenden Prozessfragmente werden analog zu dem Vorgehen in Kapitel 4.6.1.3 ermittelt. Beim Einfügen neuer Prozessfragmente in die Instanz beeinflusst der Ausführungsstatus der einzelnen Aktivitäten allerdings nicht nur ob das Fragment überhaupt eingefügt werden darf, sondern auch in welcher Weise es der Instanz hinzugefügt wird. In Kapitel 4.5.3 wurden Transformationsregeln aufgestellt, die bewirken, dass ein Prozessfragment möglichst sequentiell in Abhängigkeit der bereits existierenden vorhergehenden und nachfolgenden Aktivitäten in einen Prozess eingefügt wird. Die Möglichkeiten der Sequentialisierung der Prozessfragmente können durch das Voranschreiten der Prozessbearbeitung stärker eingeschränkt sein als zur Modellierungszeit; so ist es z. B. nicht erlaubt, ein neues Fragment sequentiell zwischen zwei bereits abgeschlossene Aktivitäten einzufügen. In diesem Fall sollte das neue Fragment jedoch baldmöglichst bearbeitet werden und sollte der Instanz daher so hinzugefügt werden, dass es sofort nach der Auswertung der Markierungsregeln den Zustand *Activated* annimmt (siehe Kapitel 4.6.2.1). Aus diesem Grund müssen für das Einfügen neuer Prozessfragmente in eine Prozessinstanz teilweise andere Transformationsregeln gelten, als diejenigen für die Transformation von Prozessdefinitionen. Diese Regeln hängen von den in Kapitel 4.5.3 definierten Mengen N_{Before}, N_{After} und $N_{Clon} \subseteq N_{Before}$ ab.

Sei $I = (W_I, \Delta_I, M^{WI}, Val^{WI}, \Pi^{WI}_I)$ eine Prozessinstanz und die Mengen N^{WI}_{Before}, N^{WI}_{After} und N^{WI}_{Clon} analog zu den Mengen in Kapitel 4.5.3 spezifiziert, dann sind die folgenden Fallunterscheidungen vorzunehmen:

- Wenn gilt $N^{WI}_{Before} = N^{WI}_{After} = \varnothing$, dann folgt $N^{WI}_{Before} := N^{WI}_{Before} \cup \{n \in N^{WI} \setminus N^{WI}_{Before} | NT^{WI}(n) \notin \{StartFlow, EndFlow\} \wedge NS^{WI}(n) \in \{Running, Completed, Failed\}\}$. Der Einfügevorgang folgt anschließend den in Kapitel 4.5.3 spezifizierten Regeln.
- Wenn gilt $N^{WI}_{After} = \varnothing$, $N^{WI}_{Before} \neq \varnothing$, dann folgt $N^{WI}_{Before} := N^{WI}_{Before} \cup \{n \in succ^*(N^{WI}_{Before}) | NT^{WI}(n) \notin \{StartFlow, EndFlow\} \wedge NS^{WI}(n) \in \{Running, Completed, Failed\}\}$. Anschließend wird das neue Prozessfragment nach den in Kapitel 4.5.3 spezifizierten Regeln in die Instanz eingefügt.
- Wenn gilt $N^{WI}_{After} \neq \varnothing$, $N^{WI}_{Before} = \varnothing$ oder $N^{WI}_{Before} \neq \varnothing$, dann folgt $N^{WI}_{Before} := N^{WI}_{Before} \cup \{n \in N^{WI} \setminus N^{WI}_{Before} | NT^{WI}(n) \notin \{StartFlow, EndFlow\} \wedge NS^{WI}(n) \in \{Running, Completed, Failed\}\}$ und $N^{WI}_{After} := N^{WI}_{After} - (N^{WI}_{Before} \cap N^{WI}_{After})$. Im Anschluss wird die Transformation gemäß den normalen Transformationsregeln durchgeführt, die auch für Prozessdefinitionen gelten.

Die nächste Abbildung verdeutlicht das Einfügen eines neuen Teilgraphen in eine Prozessinstanz im Anschluss an die Rekonfiguration des SPF-Graphen an einem Beispiel.

Abbildung 108 Einfügen von Prozessfragmenten in eine Prozessinstanz nach der Rekonfiguration des SPF-Graphen



Wie aus dem Beispiel hervorgeht, wird durch strukturerhaltende Rekonfiguration ein zweiter Klonknoten vom Typ *B* in den SPF-Graphen eingefügt. Die Menge V_{Trans} umfasst anschließend die beiden neuen Knoten *D* und *E*. Da es zwischen *D* und allen anderen Knoten im SPF-Typ-Graphen keine Abhängigkeiten gibt, beinhaltet N_{Before} lediglich die bereits ausgeführte Prozessaktivität *E*; dementsprechend kann nun das mit *D* assoziierte Prozessfragment parallel zu Aktivität *F* in die Prozessinstanz eingefügt werden. Beim Einfügen des neuen Fragments *E* umfasst N_{Before} immer noch die abgeschlossene Aktivität *E* und N_{After} enthält Aktivität *F*, da zwischen den entsprechenden Knoten im SPF-Typ-Graphen ein Sequenz-Constraint definiert ist. Aufgrund dessen wird das neue Prozessfragment sequentiell zwischen den AndSplit-Knoten $NULL_1$ und Prozessaktivität *F* in die Instanz eingefügt. Nach Bestätigung der Modifikationen werden die Knoten- und Kantenmarkierungen von ADEPT2 automatisch aktualisiert.

4.6.3 Identifikation von Knoten im SPF-Graphen zur Rekonfiguration

Während die Konfiguration einem strukturierten Vorgehen entspricht, das beim Wurzelknoten des SPF-Typ-Graphen beginnt und sich top-down orientiert über die verschiedenen Subgraphen bis hin zu den SPF Knoten erstreckt, kann bei der Rekonfiguration kein solches Muster vorausgesetzt werden. Die Selektion eines Knotens bzw. einer Knotenmenge zur Rekonfiguration kann prinzipiell beliebig erfolgen. Bei einer rein manuellen Selektion, müssten Fachanwender den gesamten bisherigen SPF-Graphen durchsuchen und diejenigen Knoten markieren, deren Konfiguration geändert werden soll. Dies entspricht jedoch einem sehr mühsamen und fehleranfälligen Vorgehen, insbesondere bei großen und komplexen Graphen. Angenommen, bei einer klinischen Behandlung wie z. B. bei der stationären Versorgung von Wirbelsäulenuntersuchungen, sollen mehrere radiologische Kontrollaufnahmen angefertigt werden. Während der Anamnese stellt die behandelnde Ärztin fest, dass der Patient über einen Herzschrittmacher verfügt; da MRT-Untersuchungen in Kombination mit Herzschrittmachern kontraindiziert sind, müsste sie nun den gesamten SPF-Graphen nach Knoten durchsuchen, die mit Prozessfragmenten für die Beauftragung und Erstellung von MRTs assoziiert sind. Ein weiteres Beispiel bezieht sich auf spezielle diagnostische oder therapeutische Maßnahmen: Angenommen, die Ärztin verordnet dem Patienten abweichend vom klinischen Pfad an den ersten drei post-operativen Tagen eine spezielle Atmungstherapie. Die Ärztin müsste die Rekonfigurationsknoten also manuell im Kontext dieser drei post-operativen Tage bestimmen. Um die Selektion von Rekonfigurationsknoten für Fachanwender zu erleichtern, wird in diesem Kapitel ein Algorithmus für die Identifikation von Knoten in SPF-Graphen entwickelt. Dieser Algorithmus lässt sich auch bei der Anwendung von Prozessvarianten nutzen (siehe Kapitel 4.7).

In Kapitel 4.4.4.2 wurde bereits erläutert, dass sich die Anwendung einer Produktionsregel entweder auf die eindeutige Identität eines Knoten im SPF-Graphen beziehen kann oder gleichzeitig auf alle Klone dieses Knoten erstreckt. Diese Zusammenhänge werden in spezifischerer Form auch bei der Identifikation von Knoten zur Rekonfiguration genutzt. Insgesamt werden die folgenden Selektionsbedingungen unterschieden:

- **Alle:** Es werden alle Knoten im SPF-Graphen selektiert, die auf denselben Knoten im SPF-Typ-Graphen abgebildet werden (z. B. wähle alle Untersuchungen).
- **Kontextsensitiv:** Die Menge der selektierten Knoten muss sich in einem definierten Kontext befinden (z. B. wähle alle Untersuchungen im Kontext Diagnostik).
- **Attributbezogen:** Die Attribute der Knoten müssen definierte Werte annehmen (z. B. wähle alle Pflegeaktivitäten, die der Mobilisation dienen; d. h. Attribut Mobilisation = »true«).
- **Prozessbezogen:** Die Auswahl der Knoten hängt von ihrer Position innerhalb der Prozessdefinition ab (z. B. wähle die ersten drei radiologischen Untersuchungen).

Dabei ist zu beachten, dass grundsätzlich die Möglichkeit besteht, kontextsensitive, attributbezogene und prozessbezogene Bedingungen miteinander zu kombinieren, um die Menge der selektierten Knoten weiter einzuschränken. Bevor in dieser Sektion der Algorithmus für die Funktion *identifyNodes* zur Identifikation von Knoten, die bestimmte Bedingungen erfüllen, spezifiziert wird, soll zuerst die Struktur des Filterelements *Conditions* definiert werden:

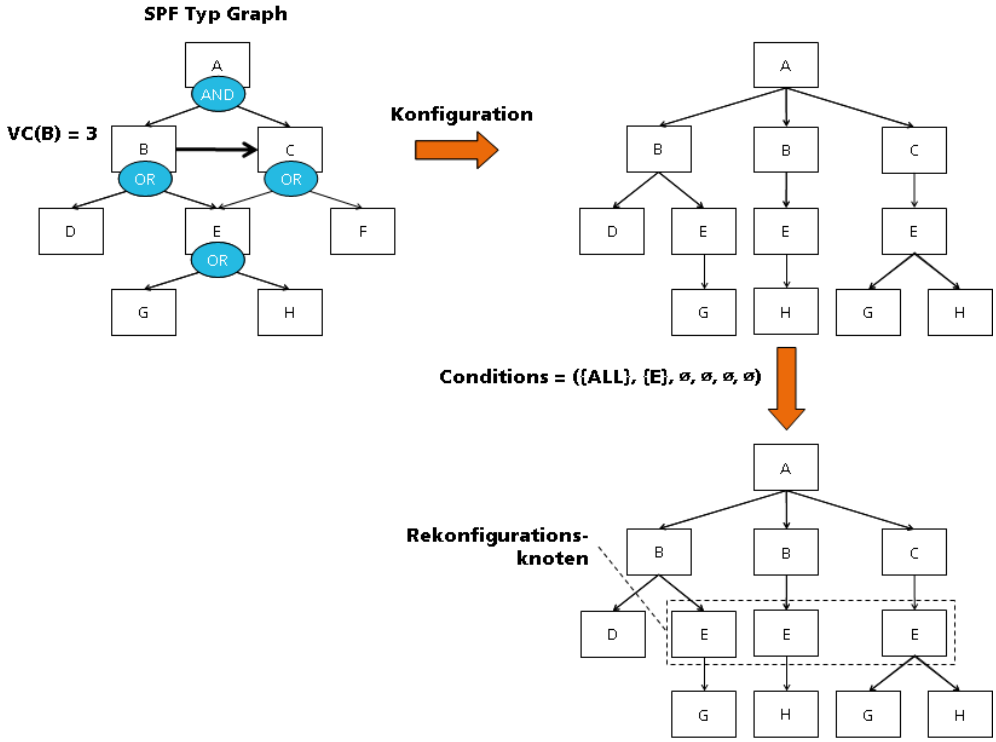
$$Conditions = (Mode^{Conditions}, V^{Conditions}, V^{Conditions}{}_{Context}, A^{Conditions}, VA^{Conditions}, Positions^{Conditions})$$

Bei $Mode^{Conditions} \subseteq \{ALL, CONTEXT, ATTRIBUTE, PROCESS\}$ handelt es sich um einen obligatorischen Parameter, mit dem der grundsätzliche Auswahlmodus festgelegt wird. $V^{Conditions}$ ist eine Menge von Knotenbezeichnungen, die den Bezeichnungen der Knoten im

SPF-Typ-Graphen entsprechen sollen; es gilt also $V^{Conditions} \subseteq V^{STG}$. Ist dies nicht der Fall, kann ein Algorithmus zur Identifikation von Knoten im davon abgeleiteten SPF-Graphen natürlich kein Ergebnis zurückliefern. Dementsprechend muss $V^{Conditions}$ auch mindestens ein Element umfassen. Die Werte der übrigen Parameter werden in Abhängigkeit des gewählten Selektionsmodus gesetzt. Ihre Beschreibung erfolgt in diesem Kapitel daher zusammen mit der Darstellung des jeweiligen Modus.

Entspricht der Modus dem Wert *ALL* so werden alle Knoten im SPF-Graphen identifiziert, deren Abbild einem Knoten aus $V^{Conditions}$ entspricht. Dieser Modus kann z. B. genutzt werden, um effizient alle Untersuchungen im SPF-Graphen zu bestimmen. Die nachfolgende Abbildung demonstriert an einem Beispiel die Auswirkung von *ALL* auf die Selektion von Rekonfigurationsknoten. In dem Beispiel umfasst die Menge der Knotenbezeichnungen $V^{Conditions}$ das Element *E*.

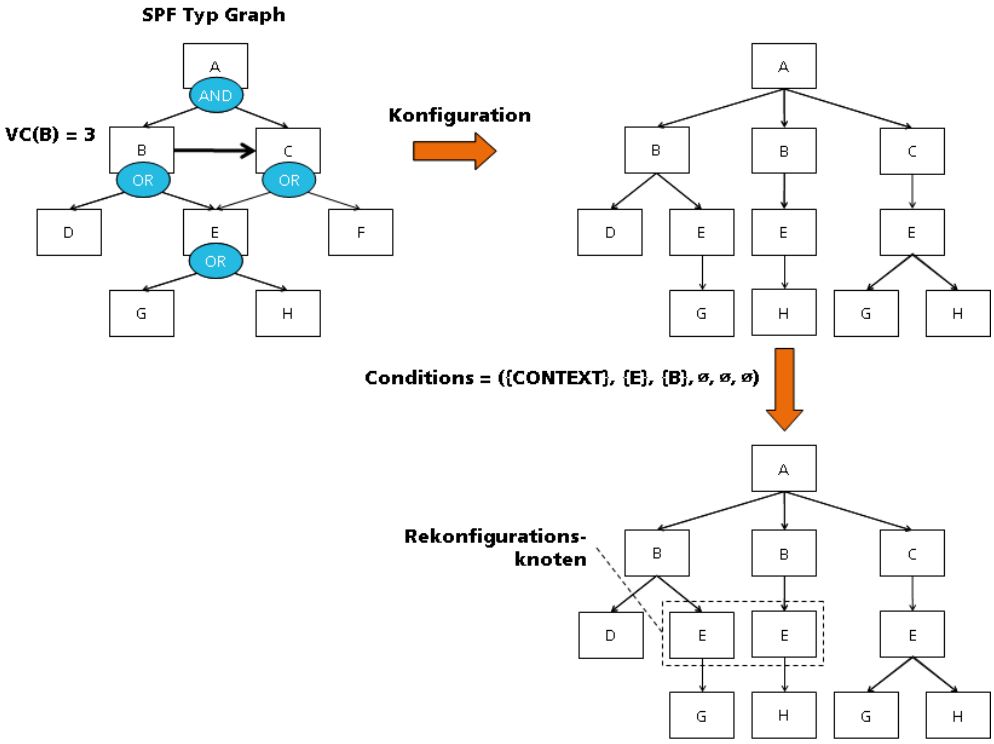
Abbildung 109 Identifikation von Rekonfigurationsknoten in Abhängigkeit ihrer Knotenbezeichnung im SPF-Typ-Graphen



Für die Identifikation von terminalen Knoten werden nun noch drei weitere Modi eingeführt. Beim Setzen des Modus auf *CONTEXT* wird eine kontextsensitive Filterung veranlasst. Dieses Verfahren ist vor allem bei Mehrfachreferenzen sinnvoll; dabei handelt es sich um Knoten mit identischem Abbild im SPF-Typ-Graphen, die jedoch im SPF-Graphen unterhalb unterschiedlicher Kontextknoten auftreten. Mit Hilfe der kontextsensitiven Filterung lässt sich also eine Untersuchung, die im Rahmen der Diagnostik stattfindet, von einer Untersuchung während der Therapie unterscheiden. Es ist jedoch auch möglich, Rekonfigurationsknoten mit unterschiedlichen Knotenbezeichnungen im SPF-Typ-Graphen zu identifizieren, die sich in demselben Kontext befinden, wie z. B. alle radiologischen und labortechnischen Untersuchungen im Kon-

text Diagnostik. Abbildung 110 verdeutlicht das Ergebnis der Filterung an einem Beispiel. Selektiert werden alle Knoten mit der Bezeichnung *E*, die sich im Kontext von *B* befinden.

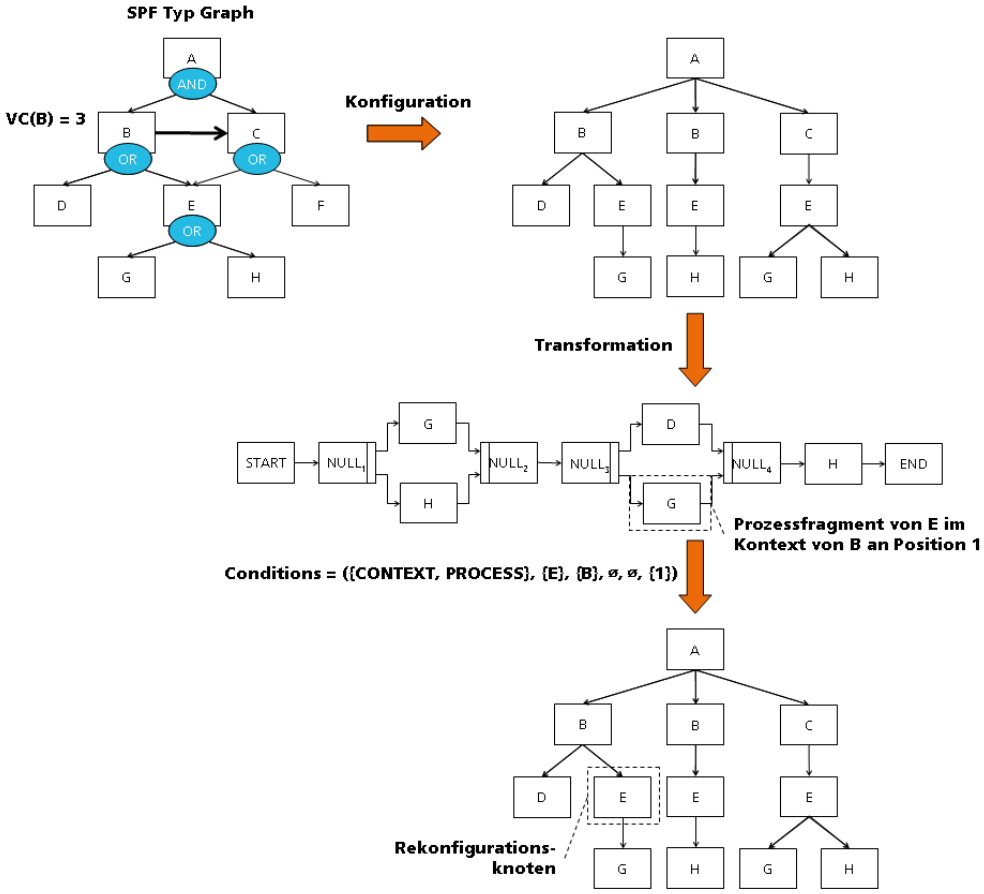
Abbildung 110 Kontextsensitive Identifikation von Rekonfigurationsknoten in Abhängigkeit ihrer Knotenbezeichnung



Der Modus *ATTRIBUTE* bewirkt die Filterung von Knoten in Abhängigkeit ihrer Attribute. Die Attribute, die bei der Suche eingesetzt werden, sollten Attributen aus dem SPF-Typ-Graphen entsprechen; d. h. $A^{Conditions} \subseteq A^{STG}$. Auch der Wertebereich muss sich mit dem dort angegebenen Wertebereich decken, damit Ergebnisse zurückgeliefert werden können. Ob der kontextsensitive oder attributbezogene Modus ausgewählt wird, hängt im Wesentlichen vom Aufbau des SPF-Typ-Graphen ab. In Kapitel 4.3.3 wurde das Anlegen von Knoten von der Spezifikation von Attributen abgegrenzt, um bestimmte Ausprägungen von Prozessfragmenten festzulegen.

Als einziger Modus hängt bei *PROCESS* die Identifikation der Knoten nicht nur von der Struktur des SPF-Graphen, sondern auch von der Anordnung der Prozessfragmente in der konkreten Prozessdefinition, also z. B. im WSM Net ab. $Positions^{Conditions}$ ist eine Menge über Positionsangaben als ganze Zahlen, also $Positions^{Conditions} \subseteq \mathbb{N}$. So können Knoten danach selektiert werden, ob sie in der Prozessdefinition an erster, zweiter, oder letzter Stelle vorkommen. Während Klone unterhalb desselben gemeinsamen Kontextknotens immer in sequentieller Reihenfolge im Prozess angeordnet sind, können Klone auf Basis von Mehrfachreferenzen auch parallel zueinander auftreten. Daher kann die Treffermenge pro gewünschte Position auch mehrere Knoten enthalten. Die nächste Abbildung illustriert den Auswahlmechanismus bei einer Kombination der Modi *PROCESS* und *CONTEXT*.

Abbildung 111 Kontextsensitive und prozessbezogene Identifikation von Knoten in Abhängigkeit ihrer Knotenbezeichnung



Im Gegensatz zu den anderen Modi muss bei der prozessbezogenen Selektion als Vorbedingung die Transformation des SPF-Graphen in eine Prozessdefinition bereits stattgefunden haben. Nur dann ist es möglich, die Position der Prozessfragmente, die einem Knoten untergeordnet sind, zu bestimmen und in Relation zueinander zu setzen. Während bei der kontextsensitiven Filterung noch beide Knoten mit Bezeichnung *E* unterhalb von *B* für die Rekonfiguration in Frage kommen, reduziert sich diese Menge durch die prozessbezogene Selektion auf den Knoten *E* im Kontext des ersten Klonknoten von *B*.

Die Erzeugung eines Objekts vom Typ *Conditions* durch Interaktionen mit Fachanwendern, hängt von der konkreten Realisierung der Anwendung für die Rekonfiguration von Prozessen ab. Da alle Konzepte zur Erfassung der Filterbedingungen nur beispielhaften Charakter hätten, wird dieser Aspekt in der Arbeit nicht behandelt. In der hier vorliegenden Dissertation spielt der Algorithmus zur Identifikation von Knoten im SPF-Graphen vor allem für die Realisierung von Prozessvarianten (siehe Kapitel 4.7) eine wichtige Rolle; aus diesem Grund bezieht er sich auch ausschließlich auf vollständige Konfigurationen von SPF-Typ-Graphen.

Das folgende Codesegment entspricht nun dem Algorithmus für die Identifikation von Knoten, die bestimmte Selektionsbedingungen erfüllen. Neben dem SPF-Typ-Graphen, dem konkreten SPF-Graphen und der daraus abgeleiteten Prozessdefinition wird der Funktion auch der Parameter *Conditions* übergeben. Die Funktion liefert alle terminalen Knoten im SPF-Graphen

zurück, die die Bedingungen erfüllen. Beim Parameter *W* handelt es sich entweder um eine Prozessdefinition oder die Kopie der Definition im Rahmen einer Prozessinstanz.

Formel 24 Algorithmus zur automatischen Identifikation von Knoten im SPF-Graphen

```
identifyNodes(STG, SG, W, Conditions) {
      V_Reconfig := ∅ ;
      //Alle terminalen Klonknoten auswählen
      V_Reconfig := {v^SG ∈ V^SG_T | ∃u ∈ V^Conditions: u = g_v(v^SG)};
      V_Reconfig := {v^SG ∈ V_Reconfig | ∄ u^SG ∈ V_Reconfig: u^SG →* v^SG};
      IF ({ALL} ∈ Mode^Conditions)
         return V_Reconfig;
      END IF
      ELSE
         IF ({CONTEXT} ∈ Mode^Conditions)
            //Terminale Knoten kontextsensitiv nach Vorgängern auswählen
            V_Reconfig := {v^SG ∈ V_Reconfig | ∀u ∈ V^conditions_Context ∃u^SG ∈ V^SG_T: u = g_v(u^SG) ∧
u^SG →* v^SG};
         END IF
         IF ({ATTRIBUTE} ∈ Mode^Conditions)
            //Terminale Knoten attributbezogen auswählen
            V_Reconfig := {v^SG ∈ V_Reconfig | ∃u ∈ V^Conditions: u = g_v(v^SG) ∧ (∀a ∈ A^conditions:
VA^conditions(a, u) = VA^SG(a, v^SG)};
         END IF
         IF ({PROCESS} ∈ Mode^Conditions ∧ W ≠ ∅ )
            V*_Reconfig := ∅ ;
            //SPF Knoten unterhalb der Rekonfigurationsknoten
            SPFNodes* := {v^SG ∈ V^SG | (∃u^SG ∈ V_Reconfig: u^SG = v^SG ∨ u^SG →* v^SG) ∧
g_v(v^SG) ∈ V^STG_SPF};
            //Prozessfragmente zu den SPF Knoten
            N^W_SPFNodes* := {n ∈ N^W | ∃v^SG ∈ SPFNodes* ∃p ∈ VP^STG(g_v(v^SG)): n = Name^p
∪ VID^SG(v^SG)};
            FOR (i := 0; i < |V_Reconfig|; i + 1)
               SPFNodes^i := {v^SG ∈ SPFNodes* | V_Reconfig[i] →* v^SG};
               N^W_SPFNodes^i := {n ∈ N^W_SPFNodes* | ∃v^SG ∈ SPFNodes^i ∃ p ∈ VP^STG(g_v(v^SG)): n
= Name^p ∪ VID^SG(v^SG)};
               //Ermittle Position an Hand der Vorgänger
               N^W_Pred := {n ∈ N^W_SPFNodes* | n ∈ pred*(N^W_SPFNodes^i)};
               SPFNodes_Pred := {v^SG ∈ SPFNodes* | ∃n ∈ N^W_Pred ∃p ∈ VP^STG(g_v(v^SG)): n
= Name^p ∪ VID^SG(v^SG)};
               V_position := {v^SG ∈ V_Reconfig | v^SG →* SPFNodes_Pred};
               position := |V_position| + 1;
               //Handelt es sich um eine Zielposition
               IF (position ∈ Positions^Conditions)
                  V*_Reconfig := V*_Reconfig ∪ V_Reconfig[i];
               END IF
            END FOR
            V_Reconfig := V*_Reconfig;
         END IF
      END ELSE
      return V_Reconfig;
}
```

Zu Beginn des Algorithmus wird die Menge $V^{Reconfig}$ ermittelt, die alle terminalen Knoten im SPF-Graphen umfasst, deren Abbild im SPF-Typ-Graphen mit einem Knoten aus $V^{Conditions}$ übereinstimmt; diese Menge wird außerdem um mögliche Nachfolgerknoten bereinigt. Entspricht der Selektionsmodus *ALL*, so kann die Menge $V_{Reconfig}$ bereits als Ergebnis zurückgeliefert werden. Nach der kontextsensitiven Filterung enthält $V_{Reconfig}$ nur noch solche Knoten, die für jeden Knoten der Menge $V^{Conditions}{}_{Context}$ einen Vorgänger im SPF-Graphen besitzen. Bei der

attributbezogenen Filterung müssen alle Attribute des Knotens mit den Attributwerten seines Pendants aus $V^{Conditions}$ übereinstimmen. Die prozessbezogene Selektion ist komplexer, da es zwar mittels SPF-Typ-Graph eine direkte Verbindung zwischen SPF Knoten und Prozessfragmenten gibt; diese Relation existiert jedoch nicht in umgekehrter Richtung. Der wesentliche Grund hierfür ist, dass SPF-Graphen bei der Transformation auf bestehende Prozessmodellierungssprachen abgebildet werden sollen, ohne dass diese Sprachen um eine Funktion erweitert werden müssen, die Aktivitäten den SPF Knoten zuordnet. Eine wechselseitige Abhängigkeit ergibt sich also nur indirekt über die Operationalisierung der Aktivitäten durch die Implementierungen der Prozessfragmente. Zu Beginn werden alle SPF Knoten unterhalb der ermittelten Knoten identifiziert. Anschließend werden die Prozessaktivitäten in der Prozessdefinition (hier als WSM Net) festgestellt, die mit den Prozessfragmenten der SPF Knoten korrespondieren. Dieser Vorgang erfolgt im Rahmen einer Schleife für jeden gefundenen Knoten auch einzeln. Anschließend ist es durch Ermittlung der Vorgängeraktivitäten und ihrer Zuordnung zu anderen Knoten möglich, die Position der Prozessfragmente des aktuellen Knotens festzustellen. Entspricht diese Position einer der in $Positions^{Conditions}$ gewünschten Positionen, so wird der aktuelle Knoten in die Ergebnismenge aufgenommen. Je nach Art der Verwaltung der Elemente der Menge $V_{Reconfig}$ lassen sich noch verfeinerte Filtermechanismen umsetzen. Werden die Elemente der Menge z. B. in einer HashMap abgelegt, kann als Schlüssel die Positionsnummer eingetragen werden; auf diese Weise ist es möglich, auf die Knoten im SPF-Graphen an Hand der Position ihrer Prozessfragmente im WSM Net zuzugreifen. So können z. B. auch erweiterte Filtermechanismen zur Bestimmung des ersten, des nächsten oder des letzten Knotens und der entsprechenden Fragmente im Prozess umgesetzt werden.

Generell gilt zu bedenken, dass der Selektionsalgorithmus nicht zwischen Prozessdefinitionen und Prozessinstanzen unterscheidet. Das bedeutet, in der zurückgelieferten Ergebnismenge können sich auch Rekonfigurationsknoten befinden, die aufgrund ihres aktuellen Ausführungsstatus gar nicht rekonfiguriert werden dürfen. Dies ist jedoch auch nicht Aufgabe des hier spezifizierten Algorithmus und wird spätestens beim Versuch der Rekonfiguration durch die in Kapitel 4.6.2.3 definierten Funktionen erkannt.

4.6.4 Zusammenfassung der Ergebnisse

Die Rekonfigurationsphase dient der Realisierung von ad hoc Änderungen an Prozessdefinitionen und ihren Instanzen. Sie basiert auf der Anpassung von SPF-Graphen, wobei sich die möglichen Änderungsoperationen auf das Hinzufügen neuer Knoten oder das Entfernen vorhandener Knoten beschränken. Die konkreten Auswirkungen der Rekonfiguration auf den Prozess ergeben sich erst durch die erneute Transformation des modifizierten Teilgraphen; diese folgt den Regeln, die bereits in Kapitel 4.5.3 spezifiziert wurden. Während die Rekonfiguration von SPF-Graphen zur Modellierungszeit vollkommen unabhängig von spezifischen Systemen dargestellt wurde, kann die Transformation von Prozessdefinitionen, aber auch die Rekonfiguration und Transformation von Prozessinstanzen nur in Abhängigkeit einer konkreten Prozessmodellierungssprache und des zugrunde liegenden WfMS erfolgen. Hier werden die Freiheiten der Rekonfiguration durch den aktuellen Ausführungsstatus der Prozessinstanz eingeschränkt. In dieser Arbeit wurde das WfMS ADEPT2 ausgewählt, um auch auf diese Aspekte des Lösungskonzepts adäquat eingehen zu können.

Bei der Betrachtung der Rekonfigurationsphase wird also generell zwischen der Rekonfiguration zur Modellierungszeit und der Rekonfiguration zur Laufzeit differenziert. In Abhängigkeit dessen wurden die Bedingungen für die Rekonfiguration definiert. Bei der Rekonfiguration zur

Modellierungszeit können prinzipiell dieselben arbeitserleichternden Methoden eingesetzt werden wie bei der normalen Konfiguration; dabei handelt es sich um den Einsatz von Algorithmen, die die Einhaltung von Constraints automatisch überprüfen und erzwingen. Grund dafür ist, dass die Rekonfiguration eines SPF-Graphen zur Modellierungszeit meist im Kontext der ursprünglichen Konfiguration stattfindet, wobei bereits vorgenommene Konfigurationsschritte zurückgenommen oder ergänzt werden müssen. Die Situation zur Laufzeit unterscheidet sich von diesem Arbeitskontext hingegen fundamental: Hier stehen Anwender unter dem Druck den bereits laufenden Prozess ad hoc zu modifizieren, ohne die bisherigen Arbeiten zu behindern oder die weitere Planung zu erschweren. Aus diesem Grund soll das Auslösen von kaskadierenden Änderungen am SPF-Graphen, das bei dem automatischen Erzwingen der Einhaltung von Constraints entstehen kann, unterdrückt werden. Sämtliche Anpassungen am SPF-Graphen zur Laufzeit stehen also vollständig unter der Kontrolle des verantwortlichen Anwenders.

Generell werden bei der Rekonfiguration von SPF-Graphen zwei Vorgehensweisen unterschieden: Die strukturerhaltende und die nicht-strukturerhaltende Rekonfiguration. Die nicht-strukturerhaltende Rekonfiguration repräsentiert die einfachere Methode, bei der ein Teilgraph komplett durch das Nichtterminalsymbol des Wurzelknotens ersetzt wird. Da dies z. B. beim Einfügen oder Löschen einzelner Nachfolger des Wurzelknotens nicht gewünscht ist, werden neben einem Algorithmus für die nicht-strukturerhaltende Rekonfiguration auch Funktionen definiert, die dem Erhalt der bisherigen Struktur des Teilgraphen dienen. Sowohl für die Rekonfiguration zur Modellierungszeit als auch für die Rekonfiguration zur Laufzeit wird das Vorgehen zur Transformation des modifizierten Subgraphen am Beispiel von ADEPT2 WSM Nets spezifiziert. Abschließend wurde ein Algorithmus definiert, der es ermöglicht, Knoten für die Rekonfiguration an Hand von Filterkriterien zu bestimmen, um so den Rekonfigurationsaufwand für die Fachanwender zu senken. Im Überblick konnten also die folgenden Ergebnisse erzielt werden:

- Spezifikation der Bedingungen und Algorithmen für die Rekonfiguration von SPF-Graphen zur Modellierungszeit unabhängig von konkreten WfMS
- Spezifikation der Bedingungen und Algorithmen für die Rekonfiguration von SPF-Graphen zur Laufzeit am Beispiel von ADEPT2 WSM Nets und deren Ausführungssemantik
- Definition der notwendigen Schritte zur Transformation von SPF-Graphen nach der Rekonfiguration für Prozessdefinitionen und –instanzen
- Darstellung eines Verfahrens und des notwendigen Algorithmus zur Identifikation von Knoten im SPF-Graphen zur Rekonfiguration

4.7 Realisierung von Prozessvarianten

Wie im vorausgehenden Kapitel zur Rekonfiguration von SPF-Graphen und Transformation von Prozessinstanzen gezeigt wurde, werden zur Laufzeit sämtliche Rekonfigurationsschritte mit Ausnahme der automatisierten Regelanwendung (siehe Kapitel 4.4.4.1) von Fachexperten kontrolliert. Das hat den Vorteil, dass der SPF-Graph nicht automatisch durch das Einfügen erforderlicher Knoten oder das Löschen Konflikt-auslösender Knoten verändert wird. Der für die Rekonfiguration verantwortliche Anwender muss also nicht alle Zusammenhänge zwischen den Prozessfragmenten überblicken, sondern wird vom System lediglich auf das Verletzen von Constraints, die im SPF-Typ-Graphen definiert wurden, aufmerksam gemacht. Mit der Abgabe von Kontrollfunktionen an den Anwender steigt jedoch auch dessen Aufwand bei der Rekonfiguration zur Laufzeit. Insbesondere im Kontext von Rekonfigurationen, die aus einer Vielzahl einzelner Rekonfigurationsoperationen bestehen, ist die rein manuelle Anpassung

des SPF-Graphen ineffizient. Berücksichtigt man darüber hinaus die Analyse der Flexibilisierungsanforderungen an klinische Pfade, bei der festgestellt wurde, dass eine Reihe von Abweichungen, z. B. basierend auf Vorerkrankungen, Komplikationen oder organisatorischen Rahmenbedingungen, häufig auftreten können und ihre Folgen für den Behandlungsprozess im Voraus abschätzbar sind, ist der für die Anwender verbundene Aufwand bei der Rekonfiguration kaum mehr zu rechtfertigen. Mit der Anzahl der Änderungen und der Häufigkeit, mit der diese Änderungen manuell für unterschiedliche Behandlungsfälle wiederholt werden müssen, steigt zudem meist die Fehlerrate. Fehler in der Behandlungsplanung beeinflussen im günstigsten Fall lediglich die Kosten der medizinischen Versorgung; im ungünstigsten Fall wirkt sie sich jedoch auch auf deren Qualität aus.

Bereits bei der Analyse der Flexibilisierungsanforderungen in Kapitel 2.2 wurde daher die Forderung erhoben, von den Details komplexer Abweichungen zu abstrahieren und diese für die Fachexperten gebündelt als Prozessvarianten bereitzustellen. Eine Prozessvariante kann sich z. B. auf die Besonderheiten bei der Behandlung von Patienten mit Diabetes mellitus oder Herzinsuffizienz beziehen, die im klinischen Pfad nicht standardmäßig berücksichtigt werden. Prozessvarianten definieren somit Standards bzw. »best practice« Vorgehensweisen im Fall einer Abweichung vom klinischen Pfad. Werden Abweichungsmaßnahmen aufgrund von Varianz vom klinischen Pfad bereits im SPF-Typ-Graphen vorgesehen, so können prinzipiell individuelle Pfade nach demselben Muster wie Standardpfade durch Konfiguration des Graphen erzeugt werden. Auf diese Weise lassen sich neben dem klinischen Pfad zur stationären Behandlung von Wirbelsäulenerkrankungen, auch spezielle Pfade erstellen, die die besonderen Bedürfnisse von Patientinnen und Patienten mit Diabetes oder Herzinsuffizienz berücksichtigen. Mit der Bereitstellung einer Vielzahl verschiedener SPF-Graphen, die ausgehend von demselben SPF-Typ-Graphen konfiguriert werden, ergeben sich jedoch ähnliche Pflege- und Weiterentwicklungsprobleme, die bereits in Kapitel 3.3.3 im Zusammenhang mit dem Multi-Modellansatz geschildert wurden. Darüber hinaus kann auch die Notwendigkeit auftreten, dass Abweichungsmaßnahmen, die auf verschiedene Arten von Varianz zurückzuführen sind, miteinander kombiniert werden können. Außerdem muss die Auswahl und Anwendung einer Prozessvariante zur Prozessmodifikation nicht nur zur Modellierungszeit sondern grundsätzlich auch zur Laufzeit möglich sein. Wie bei der Spezifikation der Umsetzung von ad hoc Abweichungen durch Rekonfiguration des SPF-Graphen, muss bei der Selektion der Prozessvariante zur Laufzeit der Ausführungsstatus beachtet werden, in dem sich die zu ändernde Prozessinstanz aktuell befindet. Aus diesem Grund muss eine Methode entwickelt werden, die es Fachanwendern erlaubt, unterschiedliche Varianten flexibel zu beliebigen Zeitpunkten auszuwählen, an die Bedürfnisse des aktuellen Behandlungsfalles anzupassen und auf den Standardpfad anzuwenden.

Grundsätzlich basieren Prozessvarianten auf einer Menge von so genannten »SPF-Änderungsoperationen«. Bevor Prozessvarianten also formal spezifiziert werden können, ist es zunächst erforderlich, die verfügbaren Operationen zur Manipulation von SPF-Graphen zu definieren. Sollen Änderungsoperationen automatisch durchgeführt werden, so ist es notwendig, die Knoten, auf die sich die Modifikation beziehen soll, ebenfalls durch das System bestimmen zu lassen; dabei wird der in Kapitel 4.6.3 spezifizierte Algorithmus zur Identifikation von Knoten genutzt. Auf Grundlage der Spezifikation von SPF-Änderungsoperationen, können nun Prozessvarianten definiert werden, die eine Menge von diesen Operationen kapseln. Die Auswahl und Anwendung von Prozessvarianten findet stets in einem bestimmten Anwendungskontext statt; die Konfiguration dieses Anwendungskontextes ist vor allem dann notwendig, wenn Prozessvarianten variabel gestaltet und verschiedene Varianten miteinander kombiniert werden können (z. B. bei multimorbiden Patienten). Dabei wird gezeigt, dass die Model-

lierung der Variabilität von Prozessvarianten und deren Konfiguration auf denselben Prinzipien basieren kann, wie die Erstellung von Prozessfamilien als SPF-Typ-Graphen und die Ableitung konkreter Prozessdefinitionen als SPF-Graphen.

4.7.1 SPF-Änderungsoperationen zur Manipulation von SPF-Graphen

Jede Prozessvariante besteht aus mindestens einer SPF-Änderungsoperation, wie z. B. dem Ersetzen aller Röntgen-Untersuchungen durch MRT-Untersuchungen bei schwangeren Patientinnen. Die Verwendung der Begriffe »Ersetzen« und »aller« legt aber bereits nahe, dass es sich bei einer SPF-Änderungsoperation wiederum um viele kleinere Einzelschritte handeln kann. In diesem Abschnitt werden nun die verschiedenen Arten von Änderungsoperationen spezifiziert und entsprechende Funktionen angegeben, die auf Algorithmen basieren, die bereits in den vorhergehenden Kapiteln definiert wurden.

Prinzipiell könnte die Betrachtung von Prozessvarianten von der Annahme ausgehen, dass solche Varianten nicht nur zur Rekonfiguration, sondern bereits während der Konfiguration, also der Erstellung konkreter Prozessdefinitionen auf Basis von Prozessfamilien, genutzt werden. In diesem Fall würden die innerhalb einer Prozessvariante zum Einsatz kommenden Änderungsoperationen also nicht nur den in Kapitel 4.6 beschriebenen Rekonfigurationsoperationen entsprechen, sondern würden auch die automatische Konfiguration von Teilen des SPF-Typ-Graphen ermöglichen. Dies deckt sich jedoch nicht mit der Analyse der Anforderungen klinischer Pfade. Bei klinischen Pfaden handelt es sich um Prozessdefinitionen, die das Standardvorgehen in einem bestimmten Behandlungsfall beschreiben. Demzufolge geht der klinische Pfad als Ergebnis aus der Konfiguration des SPF-Typ-Graphen hervor und es existiert mit ihm bereits ein SPF-Graph, dessen Knoten mit den Prozessfragmenten innerhalb der Prozessdefinition korrespondieren. Die Notwendigkeit der Abweichung vom Standard wird nur ausgehend von der Betrachtung des klinischen Pfades als vorliegende Prozessdefinition identifiziert. Somit bezieht sich eine Prozessvariante immer auf den bereits konfigurierten SPF-Typ-Graphen und dieser kann durch eine Prozessvariante zunächst ausschließlich rekonfiguriert werden. Dennoch kann sich an Prozessvarianten auch die Anforderung ergeben, dass sie die nachträgliche Konfiguration zumindest von Teilen des Graphen bewirkt. Dies gilt genau dann, wenn ein neuer nicht-terminaler Knoten in den SPF-Graphen eingefügt oder eine nicht-strukturerhaltende Rekonfiguration durchgeführt wird und der betroffene Knoten anschließend konfiguriert werden muss. Für die initiale Konfiguration des SPF-Typ-Graphen und für die nachträgliche Konfiguration eines nicht-vollständig konfigurierten Graphen ist das Konzept der Prozessvarianten in dieser Arbeit hingegen nicht vorgesehen. Prinzipiell steht jedoch nichts entgegen, das Konzept der Prozessvarianten auch dahingehend zu nutzen.

Definition (SPF-Änderungsoperation). Eine SPF-Änderungsoperation ist die Manipulation eines SPF-Graphen gemäß den Regeln des SPF-Typ-Graphen, aus denen er abgeleitet wurde. Seien *SG* und *SG** zwei SPF-Graphen, die auf demselben SPF-Typ-Graphen *STG* beruhen. Sei ferner *O* die Menge möglicher Änderungsoperationen und $\Delta \in O$ eine einzelne Operation aus dieser Menge.

- $SG[\Delta > SG^* :\Leftrightarrow \Delta$ kann eingesetzt werden kann, um *SG* zu manipulieren; das Manipulationsergebnis entspricht anschließend dem SPF-Graphen *SG**.
- $SG[\Delta_0, \Delta_1, ..., \Delta_n > SG^* :\Leftrightarrow$ Es existiert eine Menge von SPF-Graphen $SG_0, SG_1, ..., SG_{n+1}$ mit $SG_1 = SG$ und $SG_{n+1} = SG^*$, so dass gilt $SG_i[\Delta i > SG_{i+1}$ mit $i \in [0, 1, ...n]$.
- $\exists f: SG \mapsto STG \wedge \exists f^*: SG^* \mapsto STG \wedge \exists \varphi: SG \mapsto SG^* :\Leftrightarrow$ Sowohl für *SG* als auch für *SG** existiert ein Graphmorphismus auf den gemeinsamen SPF-Typ-Graphen *STG* und es gibt auch eine Abbildungsfunktion von *SG* auf *SG**.

Jede SPF-Änderungsoperation kann prinzipiell unabhängig von einem bestimmten SPF-Graphen definiert werden; faktisch orientieren sich Prozessvarianten jedoch am klinischen Pfad als dem Standardprozess und seinem SPF-Graphen. Die Änderungsoperationen referenzieren Knoten lediglich an Hand ihrer Bezeichnung im SPF-Typ-Graphen. Nach der Rekonfiguration des SPF-Graphen auf Basis einer SPF-Änderungsoperation, muss als nächstes erneut die Transformation der Prozessdefinition bzw. –instanz vorgenommen werden (siehe Kapitel 4.6.1.3 und 4.6.2.4). Wie bereits angedeutet, kann eine SPF-Änderungsoperation nicht nur das strukturerhaltende Einfügen und Löschen von Knoten im SPF-Graphen bewirken, sondern auch das Ersetzen und die nicht-strukturerhaltende Rekonfiguration. Während beim Löschen und Einfügen die Knotenmenge, auf die sich die Operation bezieht, jedes Mal neu identifiziert wird, entspricht das Ersetzen einem Vorgang, der sich auf dieselbe Knotenmenge bezieht. Es können also generell die folgenden SPF-Änderungsoperationen unterschieden werden:

- INSERT: Einfügen eines nicht-terminalen Knotens und optionale Konfiguration
- DELETE: Löschen eines Knoten und des mit ihm assoziierten Teilgraphen
- REPLACE: Ersetzen eines terminalen Knotens durch einen nicht-terminalen Knoten und optionale Konfiguration
- RECONFIG: Nicht-strukturerhaltende Rekonfiguration eines terminalen Knotens und optionale Konfiguration

Es gilt zu beachten, dass eine SPF-Änderungsoperation nicht zwingend auch die Konfiguration des nicht-terminalen Knotens beinhalten muss. Somit kann die Anwendung einer Prozessvariante neben automatisch ausführbaren Manipulationen am SPF-Graphen auch aus manuellen Arbeitsschritten bestehen. Der Bereich, auf den sich diese manuellen Anpassungen beziehen, ist durch die Position des nicht-terminalen Knoten im SPF-Graphen und die für ihn und seine Nachfolger verfügbaren Produktionsregeln allerdings exakt umrissen. Im Folgenden werden die generischen SPF-Änderungsoperationen formal spezifiziert und jeweils die Algorithmen für ihre Umsetzung skizziert. Änderungsoperationen, die im Kontext von Prozessvarianten genutzt werden, entsprechen dann konkret parametrisierten Aufrufen dieser generischen Operationen.

SPF-Änderungsoperation INSERT zum Einfügen von Knoten

Die SPF-Änderungsoperation INSERT dient dem Einfügen nicht-terminaler Knoten in einen SPF-Graphen. Sie beinhaltet die Suche der Rekonfigurationsknoten im SPF-Graphen, unterhalb derer der neue Knoten vorkommen soll und gegebenenfalls auch die Konfiguration der hinzugefügten Knoten. Bei den Rekonfigurationsknoten muss es sich nicht um Klonknoten

handeln; sie sollten jedoch allesamt die Bedingung erfüllen, dass der neu hinzuzufügende Knoten als Nachfolger ausgewählt werden kann. Darüber hinaus gelten, je nachdem ob es sich um eine Prozessdefinition oder eine Instanz handelt, dieselben Regeln, die bereits in Kapitel 4.6 zur Rekonfigurationsphase beschrieben wurden. Sind die Bedingungen für einen Rekonfigurationsknoten nicht erfüllt, so hat dies jedoch lediglich zur Folge, dass die Änderung für diesen Knoten nicht durchgeführt werden kann. Wurde eine temporäre Graphgrammatik beim Aufruf der Funktion mit übergeben, so werden die neu eingefügten Knoten, die alle dieselbe Knotenbezeichnung haben, gleichartig auf Basis dieser Grammatik konfiguriert. Ist diese Graphgrammatik vollkommen deterministisch, so sind im Rahmen dieser Änderungsoperation keine Aktionen von Seiten der menschlichen Fachanwender erforderlich. Existieren für einen nicht-terminalen Knoten hingegen mehr als eine Produktionsregel, ist eine Entscheidung durch Fachexperten nötig.

Tabelle 25 Definition der SPF-Änderungsoperation INSERT

SPF-Änderungsoperation INSERT
Diese Änderungsoperation fügt einen neuen nicht-terminalen Knoten unterhalb aller identifizierten Rekonfigurationsknoten ein.
Aufrufparameter: • SPF-Typ-Graph *STG* = *(V^{STG}, E^{STG}, ...)* repräsentiert die Prozessfamilie mit sämtlichen Knotenbezeichnungen. • *SG* = *(V^{SG}, E^{SG}, ...)* ist der zu modifizierende SPF-Graph. • *W* entspricht der Prozessdefinition. • *I* entspricht der Prozessinstanz. • $V^{STG}_{Add} \in V^{STG}$ ist die Menge der Knotenbezeichnung der einzufügenden nicht-terminalen Knoten in STG. • *Conditions* kapselt die Bedingungen zur Identifikation von Rekonfigurationsknoten (siehe Kapitel 4.6.3). • *temp-GG* $\subseteq$ *GG* $\cup \varnothing$ ist eine Teilmenge der Graphgrammatik *GG* für *STG*. Sie dient der nachträglichen Konfiguration der neu eingefügten Knoten.
Vorbedingungen: • *($W \neq \varnothing \Leftrightarrow I = \varnothing$) $\wedge$ ($I \neq \varnothing \Leftrightarrow W = \varnothing$) Wenn beide Parameter leer sind, ist die prozessbezogene Suche nach Knoten im SPF-Graphen nicht möglich.*
Nachbedingung: • *SG[spfInsert* > *SG**, wobei es sich bei *SG** um den SPF-Graphen handelt, der nach Anwendung der Änderungsoperation aus *SG* entsteht.
Verwendungsbeispiele: • Erweitere alle Laboruntersuchungen im Prozess um einen Blutzuckertest aufgrund einer Vorerkrankung des Patienten an Diabetes mellitus. • Füge wegen drohender Infektionsgefahr der Medikation ab dem OP-Tag bis zur Entlassung die Gabe eines Antibiotikums hinzu.

Der nachfolgende Codeabschnitt verdeutlicht die Umsetzung der SPF-Änderungsoperation INSERT.

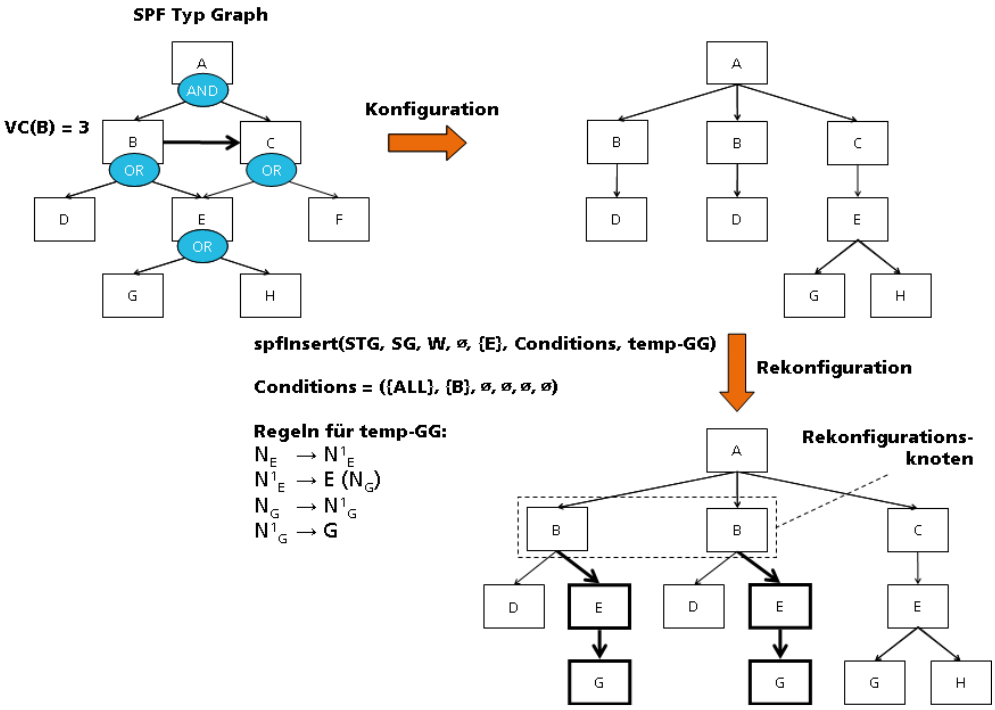
Formel 25 Algorithmus zur Umsetzung der atomaren Änderungsoperation INSERT

```
spfInsert(STG, SG, W, I, V^STG_Add, Conditions, temp-GG) {
      //Identifikation der Rekonfigurationsknoten
      IF (W ≠ ∅ )
         W^X ≔ W;
      END IF
      ELSE IF (I ≠ ∅ )
         W^X ≔ W^I;
      END ELSE
      V_Reconfig = identifyNodes(STG, SG, W^X, Conditions);
```

```
      //Knoten einfügen
      FOR (i ≔ 0; i < |V_Reconfig|; i + 1)
         IF (I ≠ ∅ )
            FOR (k ≔ 0; k < |V^STG_Add|; k + 1)
               addNodeRunTime(STG, SG, I, V_Reconfig[i], v^STG_Add[k]);
            END FOR
         END IF
         ELSE
            FOR (k ≔ 0; k < |V^STG_Add|; k + 1)
               addNodeBuildTime(STG, SG, W, V_Reconfig[i], v^STG_Add[k]);
            END FOR
         END ELSE
      END FOR
      //Knoten konfigurieren
      IF (temp-GG ≠ ∅ )
         //Neu eingefügte Knoten identifizieren
         config-nodes^SG ≔ {v ∈ V^SG_NT|source(v) ∈ V_Reconfig ∧ (∃(u, L, R) ∈ PR^temp-
GG: L = (v, ∅ , ∅ , v, v, A^SG, VA^SG, VID^SG))};;
         WHILE (|config-nodes^SG| > 0)
            pred-nodes^SG ≔ {v ∈ V^SG_T|∃config-node^SG ∈ config-nodes^SG: v →
confignode^SG};
            FOR (i ≔ 0; i < |config-nodes^SG|; i + 1)
               config-node^SG ≔ config-nodes^SG[i];
               rules ≔ {(v, L, R) ∈ PR^temp-GG|L = (config-node^SG, ∅ , ∅ , config-
node^SG, config-node^SG, A^SG, VA^SG, VID^SG))};
               rule ≔ rules[0];
               IF(|rules| = 1)
                  applyRule(temp-GG, SG, config-node^SG, rule);
               END IF
            END FOR
            t-config-nodes^SG ≔ {v ∈ V^SG_T|source(v) ∈ pred-nodes^SG};
            config-nodes^SG ≔ {v ∈ V^SG_NT|source(v) ∈ t-config-nodes^SG ∧ (∃(u,
L, R) ∈ PR^temp-GG: L = (v, ∅ , ∅ , v, v, A^SG, VA^SG, VID^SG))};
         END WHILE
      END IF
}
```

Nach der Identifikation der Rekonfigurationsknoten unterhalb derer der neue Knoten in den SPF-Graphen eingefügt werden soll, erfolgt der Einfügevorgang selbst durch Aufruf der Funktion *addNodeBuildTime* bzw. *addNodeRunTime* bei Prozessinstanzen. Dieser Algorithmus berücksichtigt nicht, ob überhaupt Rekonfigurationsknoten gefunden werden konnten oder ob das Hinzufügen der neuen Knoten erfolgreich war oder nicht. Es ist jedoch möglich, solche Ereignisse abzufangen und mit Ausnahmebehandlungen zu verknüpfen. Abschließend können die neu eingefügten Knoten mit Hilfe der zur Verfügung gestellten, temporären Graphgrammatik konfiguriert werden. Die automatische Konfiguration ist nur dann möglich, wenn für einen nicht-terminalen Knoten keine unterschiedlichen Produktionsregeln gewählt werden können; ist dies doch der Fall, müssen Fachanwender nach Abschluss der INSERT Operation selbst entscheiden, auf welche Weise die neuen Knoten konfiguriert werden sollen. Die nachfolgende Abbildung demonstriert die Auswirkungen der INSERT Änderungsoperation beispielhaft.

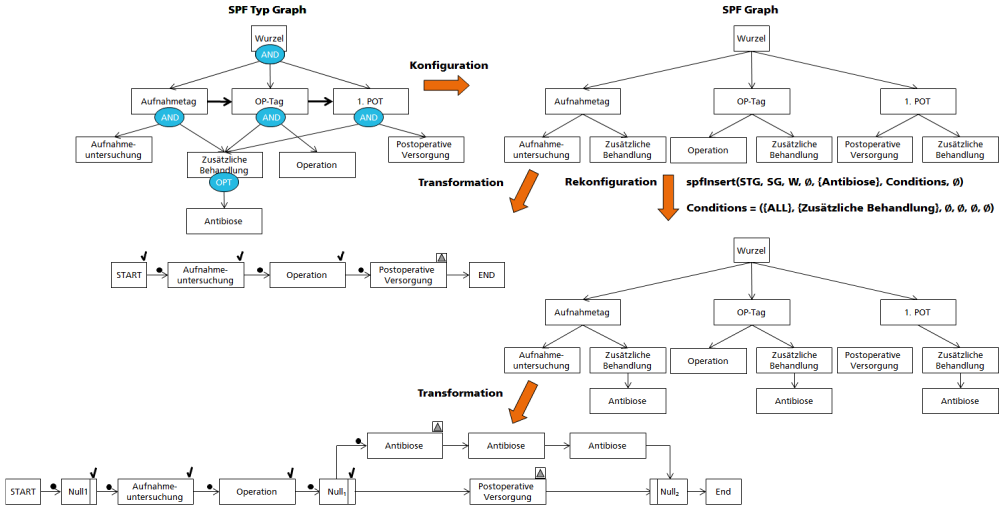
Abbildung 112 Beispiel für die Anwendung der SPF-Änderungsoperation INSERT auf einen SPF-Graphen



Gemäß Parameter *Conditions* werden innerhalb des SPF-Graphen alle Knoten *B* ausgewählt. Unterhalb jedes Rekonfigurationsknotens *B* wird anschließend der nicht-terminale Knoten N_E eingefügt. Dieser Knoten wird auf Basis der Produktionsregeln von *temp-GG* konfiguriert. Da *temp-GG* vollständig deterministisch ist, kann die Konfiguration komplett automatisch durchgeführt werden.

Bei der Auswahl der Rekonfigurationsknoten, unterhalb deren der neue Knoten hinzugefügt werden soll, gilt es jedoch zu beachten, dass der Einfügevorgang auch bei Prozessinstanzen zu jedem Zeitpunkt möglich ist. Das ist notwendig, um die Flexibilität bei der Durchführung von Abweichungsmaßnahmen im Hinblick auf den Beobachtungszeitpunkt zu maximieren. Wenn dies nicht adäquat berücksichtigt wird, kann es zu unerwünschten Effekten kommen. Angenommen, ein SPF-Typ-Graph ist grob in die Phasen »Aufnahmetag«, »OP-Tag«, »1. POT« unterteilt. Über Mehrfachreferenz ist jedem dieser Kontextknoten ein Nachfolger zugeordnet, der das spontane Einfügen einer Antibiose-Aktivität ermöglicht, für den Fall, dass während der stationären Behandlung eine Infektion auftritt. Wird nun am ersten postoperativen Tag eine Infektion diagnostiziert und daraufhin über den Aufruf der SPF-Änderungsoperation INSERT an jedem Tag im SPF-Graphen eine Antibiose eingefügt, würden in der Prozessdefinition mit einem Mal drei Antibiose-Aktivitäten erscheinen, die hintereinander ausgeführt werden sollen. Die nachfolgende Abbildung verdeutlicht dieses Problem.

Abbildung 113 Unerwünschter Effekt beim Einfügen neuer Prozessfragmente in laufende Prozessinstanzen



Es existieren unterschiedliche Möglichkeiten, solche Effekte zu vermeiden. So kann die Auswahl der Rekonfigurationsknoten kontextsensitiv erfolgen; im obigen Beispiel müsste im *Conditions*-Parameter lediglich der Knoten »1. POT« als zu beachtender Kontextknoten angegeben werden. Damit die Änderung also in Abhängigkeit des Beobachtungszeitpunkts der Infektion durchgeführt werden kann, müsste der *Conditions*-Parameter dynamisch vor Aufruf der SPF-Änderungsoperation festgelegt werden. Eine Alternative besteht in der Nutzung der Knotenattribute. Ein spezielles Attribut könnte auf den Abarbeitungsstand aller Prozessfragmente in einem bestimmten Kontext hinweisen. Dieses Attribut wird nach jeder Statusänderung der entsprechenden Aktivitäten in der Prozessinstanz aktualisiert. Bei einer attributbezogenen Selektion der Rekonfigurationsparameter wird dann überprüft, ob die Prozessfragmente im Kontext des Aufnahme-, OP- und des postoperativen Tags bereits abgeschlossen sind. Nur bei den Kontextknoten, bei denen mindestens ein Fragment noch nicht ausgeführt wurde, wird der neue Knoten zur Antibiose als zusätzliche Behandlung angehängt.

SPF-Änderungsoperation DELETE zum Löschen von Knoten

Ebenso wie neue nicht-terminale Knoten in den SPF-Graphen eingefügt und dort konfiguriert werden können, können Knoten auch wieder aus dem SPF-Graphen gelöscht werden. Dies geschieht unter Nutzung der SPF-Änderungsoperation DELETE. Die Knoten, die mit Hilfe der Funktion *identifyNodes* bestimmt werden, sind zugleich auch die zu löschenden Knoten.

Tabelle 26 Definition der SPF-Änderungsoperation DELETE

SPF-Änderungsoperation DELETE
Diese Änderungsoperation löscht alle identifizierten Knoten aus einem SPF-Graphen.
Aufrufparameter: • SPF-Typ-Graph $STG = (V^{STG}, E^{STG}, ...)$ repräsentiert die Prozessfamilie mit sämtlichen Knotenbezeichnungen. • $SG = (V^{SG}, E^{SG}, ...)$ ist der zu modifizierende SPF-Graph. • *W* entspricht der Prozessdefinition. • *I* entspricht der Prozessinstanz. • *Conditions* kapselt die Bedingungen zur Identifikation von Rekonfigurationsknoten einschließlich den Bezeichnungen der zu löschenden Knoten (siehe Kapitel 4.6.3).
Vorbedingungen:

• *(W ≠ ∅ ⇔ I = ∅) ∧ (I ≠ ∅ ⇔ W = ∅)* Wenn beide Parameter leer sind, ist die prozessbezogene Suche nach Knoten im SPF-Graphen nicht möglich.
Nachbedingung: • *SG[spfDelete > SG**, wobei es sich bei *SG** um den SPF-Graphen handelt, der nach Anwendung der Änderungsoperation aus *SG* entsteht.
Verwendungsbeispiele: • Entferne den letzten postoperativen Tag aufgrund einer möglichen Verweildauerverkürzung. • Entferne alle radiologischen Untersuchungen am Aufnahmetag, da die Bilder bereits vorliegen.

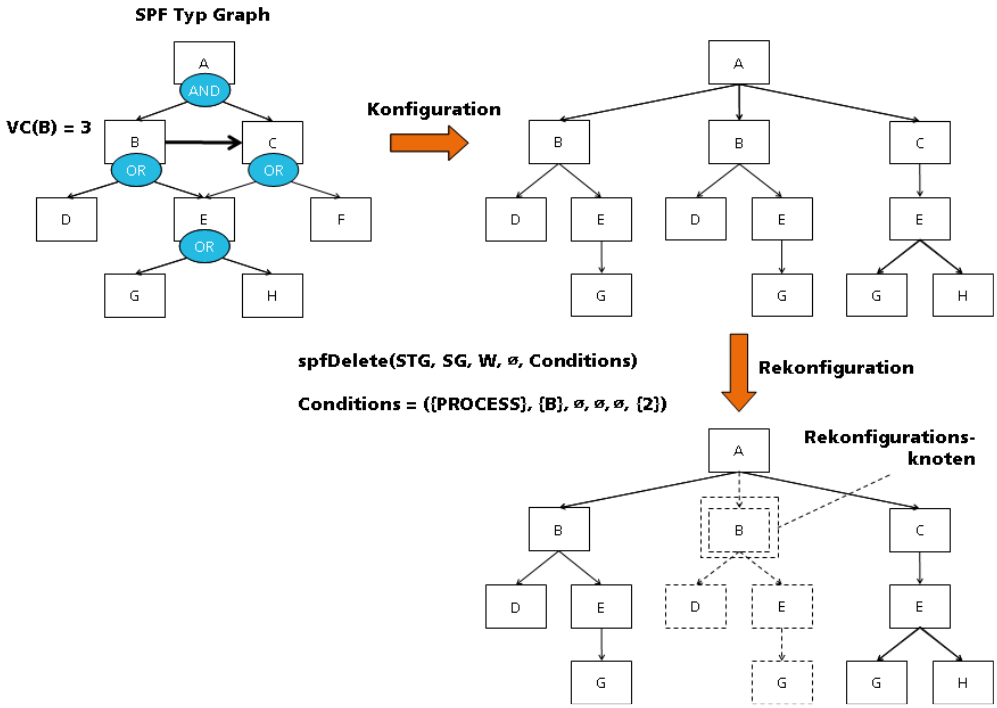
Der nachfolgende Codeabschnitt verdeutlicht die Umsetzung der SPF-Änderungsoperation DELETE.

Formel 26 Algorithmus zur Umsetzung der SPF-Änderungsoperation DELETE

```
spfDelete(STG, SG, W, I, Conditions) {
      //Identifikation der Rekonfigurationsknoten
      IF (W ≠ ∅ )
         W^X ≔ W;
      END IF
      ELSE IF (I ≠ ∅ )
         W^X ≔ W^I;
      END ELSE
      V_Reconfig ≔ identifyNodes(STG, SG, W^X, Conditions);
      //Knoten löschen
      FOR (i ≔ 0; i < |V_Reconfig|; i + 1)
         v^SG_Del ≔ V_Reconfig[i];
         pred-node^SG ≔ source(v^SG_Del);
         IF (I ≠ ∅ )
            deleteNodeRunTime(STG, SG, W, pred-node^SG, v^SG_Del);
         END IF
         ELSE
            deleteNodeBuildTime(STG, SG, I, pred-node^SG, v^SG_Del);
         END ELSE
      END FOR
}
```

Nach der Identifikation der zu löschenden Knoten im SPF-Graphen, werden sie mit Hilfe der Funktionen *deleteNodeBuildTime* bzw. *deleteNodeRunTime* zusammen mit ihren Nachfolgern aus dem Graphen entfernt. Die nachfolgende Abbildung verdeutlicht die Funktionsweise des Algorithmus an einem Beispiel.

Abbildung 114 Beispiel für die Anwendung der SPF-Änderungsoperation DELETE auf einen SPF-Graphen



Zunächst wird der Klon von *B* identifiziert, der sich in der Prozessdefinition an zweiter Position befindet. Anschließend wird der gesamte Teilgraph aus dem SPF-Graphen entfernt.

SPF-Änderungsoperation REPLACE zum Ersetzen von Knoten

Die SPF-Änderungsoperation zum Ersetzen eines Knoten im SPF-Graphen lässt sich nicht immer durch das Löschen und Einfügen von Knoten nachbilden. Die beim Löschen mit Hilfe der Funktion *identifyNodes* identifizierten Knoten werden direkt entfernt, so dass beim Einfügen unter Umständen nicht mehr klar ist, an welcher Stelle, die zuvor gelöschten Knoten im SPF-Graphen vorkamen. Aus diesem Grund wird nun die Änderungsoperation REPLACE eingeführt.

Tabelle 27 Definition der SPF-Änderungsoperation REPLACE

SPF-Änderungsoperation REPLACE
Diese Änderungsoperation dient dem Löschen und Einfügen von Knoten unterhalb derselben Kontextknoten.
Aufrufparameter: • SPF-Typ-Graph $STG = (V^{STG}, E^{STG}, ...)$ repräsentiert die Prozessfamilie mit sämtlichen Knotenbezeichnungen. • $SG = (V^{SG}, E^{SG}, ...)$ ist der zu modifizierende SPF-Graph. • *W* entspricht der Prozessdefinition. • *I* entspricht der Prozessinstanz. • $V^{STG}_{Add} \subseteq V^{STG}$ ist die Menge der Knotenbezeichnungen der einzufügenden nichtterminalen Knoten in STG. • *Conditions* kapselt die Bedingungen zur Identifikation von Rekonfigurationsknoten, einschließlich den Bezeichnungen der zu löschenden Knoten (siehe Kapitel 4.6.3). • *temp-GG* $\subseteq GG \cup \varnothing$ ist eine Teilmenge der Graphgrammatik *GG* für *STG*. Sie dient der nachträglichen Konfiguration der neu eingefügten Knoten.

Vorbedingung: • *(W ≠ ∅ ⇔ I = ∅) ∧ (I ≠ ∅ ⇔ W = ∅)* Wenn beide Parameter leer sind, ist die prozessbezogene Suche nach Knoten im SPF-Graphen nicht möglich.
Nachbedingung: • *SG[spfReplace* > *SG**, wobei es sich bei *SG** um den SPF-Graphen handelt, der nach Anwendung der Änderungsoperation aus *SG* entsteht.
Verwendungsbeispiele: • Ersetze alle Röntgenuntersuchungen durch MRT Untersuchungen. • Ersetze den Schnelltest am Aufnahmetag durch einen Blutzuckertagesspiegel und einen Blutzuckerbelastungstest.

Der nachfolgende Codeabschnitt verdeutlicht die Umsetzung der Änderungsoperation REPLACE.

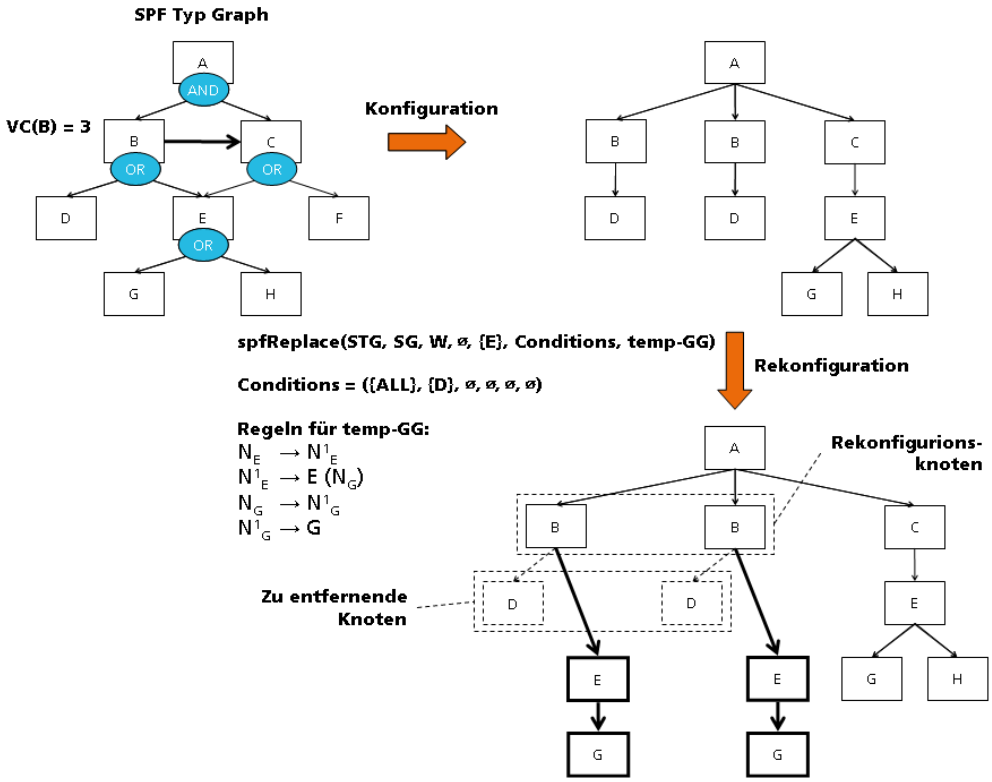
Formel 27 Algorithmus zur Umsetzung der SPF-Änderungsoperation REPLACE

```
spfReplace(STG, SG, W, I, V^STG_Add, Conditions, temp-GG) {
      //Identifikation der Rekonfigurationsknoten
      IF (W ≠ ∅ )
         W^X ≔ W;
      END IF
      ELSE IF (I ≠ ∅ )
         W^X ≔ W^I;
      END ELSE
      V_Reconfig = identifyNodes(STG, SG, W^X, Conditions);
      //Rekonfigurationsknoten merken
      pred-nodes^SG = {v ∈ V^SG|v ∈ source(V_Reconfig)};
      //Knoten löschen
      delete(STG, SG, W, I, Conditions);
      //Knoten einfügen
      FOR (i ≔ 0; i < |pred-nodes^SG|; i + 1)
         IF (I ≠ ∅ )
            FOR (k ≔ 0; k < |V^STG_Add|; k + 1)
               addNodeRunTime(STG, SG, I, pred-nodes^SG[i], v^STG_Add[k]);
            END FOR
         END IF
         ELSE
            FOR (k ≔ 0; k < |V^STG_Add|; k + 1)
               addNodeBuildTime(STG, SG, W, pred-nodes^SG[i], v^STG_Add[k]);
            END FOR
         END ELSE
      END FOR
      //Knoten konfigurieren
      IF (temp-GG ≠ ∅ )
         //Neu eingefügte Knoten identifizieren
         config-nodes^SG ≔ {v ∈ V^SG_NT|source(v) ∈ pred-nodes^SG ∧ (∃(u, L, R) ∈
PR^temp-GG: L = (v, ∅ , ∅ , v, v, A^SG, VA^SG, VID^SG))};;
         WHILE (|config-nodes^SG| > 0)
            pred-nodes^SG ≔ {v ∈ V^SG_T|∃config-node^SG ∈ config-nodes^SG: v →
confignode^SG};
            FOR (i ≔ 0; i < |config-nodes^SG|; i + 1)
               config-node^SG ≔ config-nodes^SG[i];
               rules ≔ {(v, L, R) ∈ PR^temp-GG|L = (config-node^SG, ∅ , ∅ , config-
node^SG, config-node^SG, A^SG, VA^SG, VID^SG))};
               rule ≔ rules[0];
               IF(|rules| = 1)
                  applyRule(temp-GG, SG, config-node^SG, rule);
               END IF
            END FOR
            t-config-nodes^SG ≔ {v ∈ V^SG_T|source(v) ∈ pred-nodes^SG};
            config-nodes^SG ≔ {v ∈ V^SG_NT|source(v) ∈ t-config-nodes^SG ∧ (∃(u,
```

```
L, R) ∈ PR^temp-GG: L = (v, ∅ , ∅ , v, v, A^SG, VA^SG, VID^SG))};
        END WHILE
      END IF
}
```

Nach dem Ermitteln der zu löschenden Knoten im SPF-Graphen werden deren übergeordneten Kontextknoten *pred-nodes*^{SG} als Rekonfigurationsknoten für das spätere Einfügen vorgemerkt. Dann werden die Knoten, die an Hand des Parameters *Conditions* identifiziert werden konnten, aus dem Graphen entfernt. Anschließend erfolgt das Einfügen und Konfigurieren der neuen Knoten wie bei der SPF-Änderungsoperation INSERT; allerdings werden die Vorgänger der gelöschten Knoten als Rekonfigurationsknoten verwendet.

Abbildung 115 Beispiel für die Anwendung der SPF-Änderungsoperation REPLACE auf einen SPF-Graphen



In dem obigen Beispiel wird Knoten *D* durch einen Teilgraph bestehend aus *E* und Nachfolger *G* ersetzt. Nach der Bestimmung der zu löschenden Knoten, werden die Rekonfigurationsknoten, nämlich beide Klone von *B*, für den darauf folgenden Einfügeprozess vorgemerkt. Sobald die Klonknoten von *D* aus dem SPF-Graphen gelöscht worden sind, wird unterhalb der Klone von *B* das Nichtterminalsymbol von *E* eingefügt; dieses kann auf Grundlage der temporären Graphgrammatik automatisch konfiguriert werden.

Änderungsoperation RECONFIG zur nicht-strukturerhaltenden Rekonfiguration

Mit der SPF-Änderungsoperation RECONFIG wird die nicht-strukturerhaltende Rekonfiguration einer Menge von Knoten ermöglicht. Sofern der Änderungsoperation eine temporäre Graphgrammatik als Teilmenge der regulär aus dem SPF-Typ-Graphen abgeleiteten Gramma-

tik mit übergeben wird, können die entstandenen nicht-terminalen Knoten anschließend neu konfiguriert werden. Bei den zu rekonfigurierenden Knoten muss es sich nicht gezwungenermaßen um Klonknoten handeln, da die temporäre Grammatik auch Produktionsregeln für unterschiedliche Teilgraphen umfassen kann. Bei RECONFIG handelt es sich also um eine sehr mächtige SPF-Änderungsoperation, da große Teile des SPF-Graphen komplett umstrukturiert werden können. Im Hinblick auf Prozessinstanzen ist die Nutzbarkeit der Änderungsoperation allerdings eingeschränkt, da die Voraussetzung für eine nicht-strukturerhaltende Rekonfiguration darin besteht, dass sich kein mit dem Teilgraphen assoziiertes Prozessfragment in Ausführung befindet oder bereits abgeschlossen ist.

Tabelle 28 Definition der SPF-Änderungsoperation RECONFIG

SPF-Änderungsoperation RECONFIG
Diese Änderungsoperation ermöglicht die strukturerhaltende Rekonfiguration von Knoten im SPF-Graph.
Aufrufparameter: • SPF-Typ-Graph *STG* = *(V^{STG}, E^{STG}, ...)* repräsentiert die Prozessfamilie mit sämtlichen Knotenbezeichnungen. • *SG* = *(V^{SG}, E^{SG}, ...)* ist der zu modifizierende SPF-Graph. • *W* entspricht der Prozessdefinition. • *I* entspricht der Prozessinstanz. • *Conditions* kapselt die Bedingungen zur Identifikation von Rekonfigurationsknoten, einschließlich den Bezeichnungen der zu rekonfigurierenden Knoten (siehe Kapitel 4.6.3). • *temp-GG* $\subseteq$ *GG* $\cup \varnothing$ ist eine Teilmenge der Graphgrammatik *GG* für *STG*. Sie dient der Rekonfiguration der nicht-terminalen Knoten.
Vorbedingung: • *($W \neq \varnothing \Leftrightarrow I = \varnothing$) $\wedge$ ($I \neq \varnothing \Leftrightarrow W = \varnothing$)* Wenn beide Parameter leer sind, ist die prozessbezogene Suche nach Knoten im SPF-Graphen nicht möglich.
Nachbedingung: • *SG[spfReconfig* > *SG**, wobei es sich bei *SG** um den SPF-Graphen handelt, der nach Anwendung der Änderungsoperation aus *SG* entsteht.
Verwendungsbeispiele: • Komplette Umstrukturierung des ersten post-operativen Tags aufgrund einer Verlegung in die Intensivstation. • Planungsänderung aufgrund des Versterbens des Patienten während der Operation.

Der nachfolgende Codeabschnitt verdeutlicht die Umsetzung der Änderungsoperation RECONFIG.

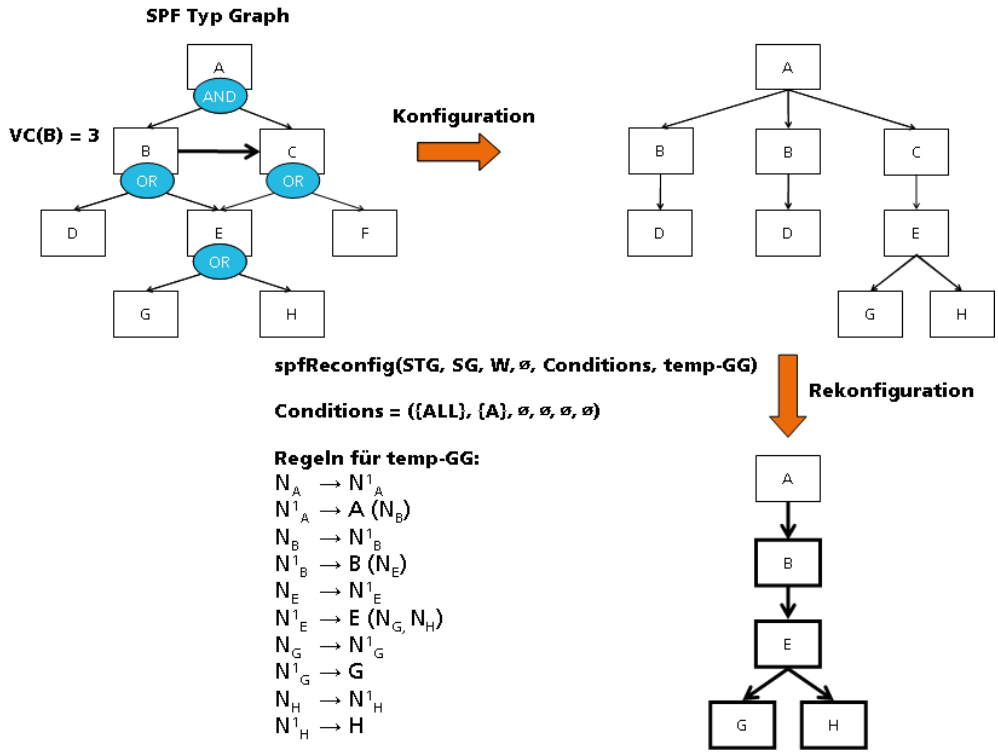
Formel 28 Algorithmus zur Umsetzung der SPF-Änderungsoperation RECONFIG

```
spfReconfig(STG, SG, W, I, Conditions, temp-GG) {
      //Identifikation der Rekonfigurationsknoten
      IF (W ≠ ∅ )
         W^X ≔ W;
      END IF
      ELSE IF (I ≠ ∅ )
         W^X ≔ W^I;
      END ELSE
      V_Reconfig = identifyNodes(STG, SG, W^X, Conditions);
      //Übergeordnete Kontextknoten für Konfiguration vormerken
      pred-nodes^SG = source(V_Reconfig);
      //Knoten nicht-strukturerhaltend rekonfigurieren
      IF (I ≠ ∅ )
         FOR (i = 0; i < |V_Reconfig|; i + 1)
            reconfigureNodeRunTime(STG, SG, I, V_Reconfig[i]);
```

```
        END FOR
      END IF
      ELSE
         FOR (i = 0; i < |V_Reconfig|; i + 1)
            reconfigureNodeBuildTime(STG, SG, W, V_Reconfig[i]);
         END FOR
      END ELSE
      //Knoten konfigurieren
      IF (temp-GG ≠ ∅ )
         //Neu eingefügte Knoten identifizieren
         config-nodes^SG ≔ {v ∈ V^SG_NT|source(v) ∈ pred-nodes^SG ∧ (∃(u, L, R)
∈ PR^temp-GG: L = (v, ∅ , ∅ , v, v, A^SG, VA^SG, VID^SG))};;
         WHILE (|config-nodes^SG| > 0)
            pred-nodes^SG ≔ {v ∈ V^SG_T|∃config-node^SG ∈ config-nodes^SG: v →
confignode^SG};
            FOR (i ≔ 0; i < |config-nodes^SG|; i + 1)
               config-node^SG ≔ config-nodes^SG[i];
               rules ≔ {(v, L, R) ∈ PR^temp-GG|L = (config-nodes^SG, ∅ , ∅ ,
config-nodes^SG, config-nodes^SG, A^SG, VA^SG, VID^SG))};
               rule ≔ rules[0];
               IF(|rules| = 1)
                  applyRule(temp-GG, SG, config-node^SG, rule);
               END IF
            END FOR
            t-config-nodes^SG ≔ {v ∈ V^SG_T|source(v) ∈ pred-nodes^SG};
            config-nodes^SG ≔ {v ∈ V^SG_NT|source(v) ∈ t-config-nodes^SG ∧ (∃(u,
L, R) ∈ PR^temp-GG: L = (v, ∅ , ∅ , v, v, A^SG, VA^SG, VID^SG))};
         END WHILE
      END IF
}
```

Nach der Identifikation der Knoten gemäß den in *Conditions* spezifizierten Bedingungen, werden die Teilgraphen auf das nicht-terminale Symbol des Wurzelknotens zurückgesetzt. Anschließend können die Nichtterminalknoten konform zu den Regeln der temporären Grammatik konfiguriert werden. Die nachfolgende Abbildung stellt den Vorgang beispielhaft dar.

Abbildung 116 Beispiel für die Anwendung der SPF-Änderungsoperation RECONFIG auf einen SPF-Graphen



Die SPF-Änderungsoperation zur nicht-strukturerhaltenden Rekonfiguration bezieht sich direkt auf den Wurzelknoten *A* und verändert die Struktur des SPF-Graphen grundlegend.

4.7.2 Formale Spezifikation von Prozessvarianten

Wie die Anforderungsanalyse in Kapitel 2.2 zur Umsetzung klinischer Pfade gezeigt hat, sind oft mehrere Änderungsoperationen notwendig, um z. B. Vorerkrankungen wie Diabetes mellitus oder Komplikationen wie Thromboembolie adäquat zu berücksichtigen. Das Zusammenfassen einzelner Änderungsoperationen führt zur Entwicklung von Prozessvarianten.

Definition (Prozessvariante). Eine Prozessvariante ist eine geordnete Menge von SPF-Änderungsoperationen, die in sequentieller Reihenfolge ausgeführt werden, um einen SPF-Graphen zu manipulieren. Seien $\Delta_0, \Delta_1, ..., \Delta_n \in O$ SPF-Änderungsoperationen; dann ist eine Prozessvariante $pv = (\Delta_0, \Delta_1, ..., \Delta_n, \hookleftarrow)$ eine geordnete Menge über diese Änderungsoperationen mit der Ordnungsrelation $\hookleftarrow :\Leftrightarrow$ *folgt auf*, d.h. die Anwendung der SPF-Änderungsoperation Δ_n folgt auf Δ_{n-1} usw.

Jede Prozessvariante, die von einem Fachexperten ausgewählt und auf einen SPF-Graphen angewendet wird, wird als Transaktion durchgeführt, die die so genannten ACID-Eigenschaften für Transaktionen erfüllt; diese Eigenschaften gelten in derselben Art und Weise auch für jede ihrer SPF-Änderungsoperationen:

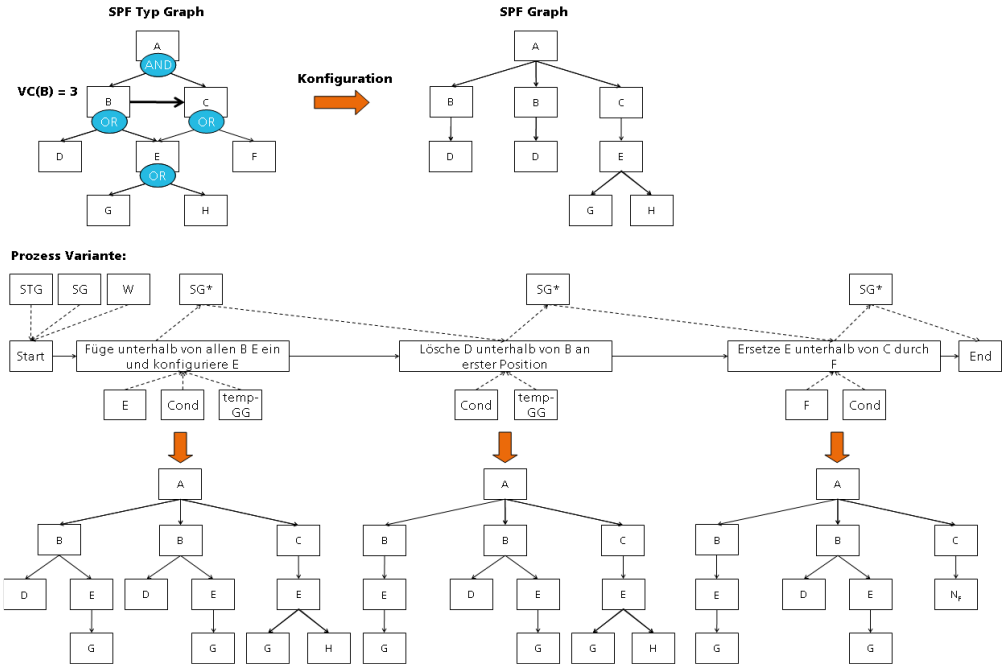
- **Atomarität**: Eine Prozessvariante wird entweder vollständig oder gar nicht ausgeführt. Gegenüber dem Fachexperten stellt sich die Prozessvariante daher wie eine einzige Operation dar. Dennoch können die einzelnen SPF-Änderungsoperationen, die von einer Variante

gekapselt werden, auch zu keinen Änderungen am SPF-Graphen führen, z. B. weil innerhalb des Graphen keine Rekonfigurationsknoten gefunden wurden, die die spezifizierten Bedingungen erfüllen.

- **Konsistenz:** Jede Prozessvariante besteht aus einer Menge von SPF-Änderungsoperationen, die auf einen SPF-Graphen angewendet werden, der konform ist zu einem gegebenen SPF-Typ-Graphen. Das Ergebnis der Manipulation ist wiederum ein SPF-Graph, der ebenfalls den Vorgaben desselben SPF-Typ-Graphen entsprechen muss. Die Konsistenz einer Prozessvariante und ihrer SPF-Änderungsoperationen muss also auf Grundlage des SPF-Typ-Graphen hergestellt sein.
- **Isolation:** Es ist nicht zulässig, dass mehr als eine Prozessvariante gleichzeitig ausgeführt wird, da Varianten und ihre Änderungsoperationen häufig nicht kommutativ sind und sich gegenseitig beeinflussen können.
- **Dauerhaftigkeit:** Der Effekt einer Prozessvariante auf einen SPF-Graphen ist dauerhaft; das Rückgängigmachen von Teilen der Variante ist nur durch die Anwendung anderer Prozessvarianten oder manueller Rekonfigurationen am SPF-Graphen möglich.

Prozessvarianten können als Anwendungsprogramme umgesetzt werden; sie können jedoch auch selbst wieder als Prozessdefinitionen realisiert sein. Jede Prozessaktivität ist dann durch eine einzelne SPF-Änderungstransaktion operationalisiert. Alternativ kann eine Aktivität auch durch eine Prozessvariante implementiert sein, was dem Schachteln von Transaktionen entspricht. Die Umsetzung von Prozessvarianten als Prozessdefinitionen hat den Vorteil, dass die Ablauflogik von der Anwendungslogik getrennt ist und einzelne SPF-Änderungsoperationen bei Bedarf effizient gegeneinander ausgetauscht werden können. Bei der Instanziierung einer Prozessvariante muss ihr zunächst die Referenz auf den zu manipulierenden SPF-Graphen übergeben werden. Die sequentielle Abarbeitung der Prozessaktivitäten entspricht dann der Durchführung von Änderungsoperationen an diesem Graphen. Die nächste Abbildung illustriert dieses Vorgehen an einem Beispiel. Aus Gründen der Übersichtlichkeit sind die Datenflüsse nicht vollständig dargestellt; mit der Zuordnung eines Datenobjekts zu einer Prozessaktivität kann jedoch davon ausgegangen werden, dass dieses Objekt im Datenkontext des WfMS auch für die nachfolgenden Aktivitäten zur Verfügung steht.

Abbildung 117 Anwendung einer Prozessvariante auf einen SPF-Graphen

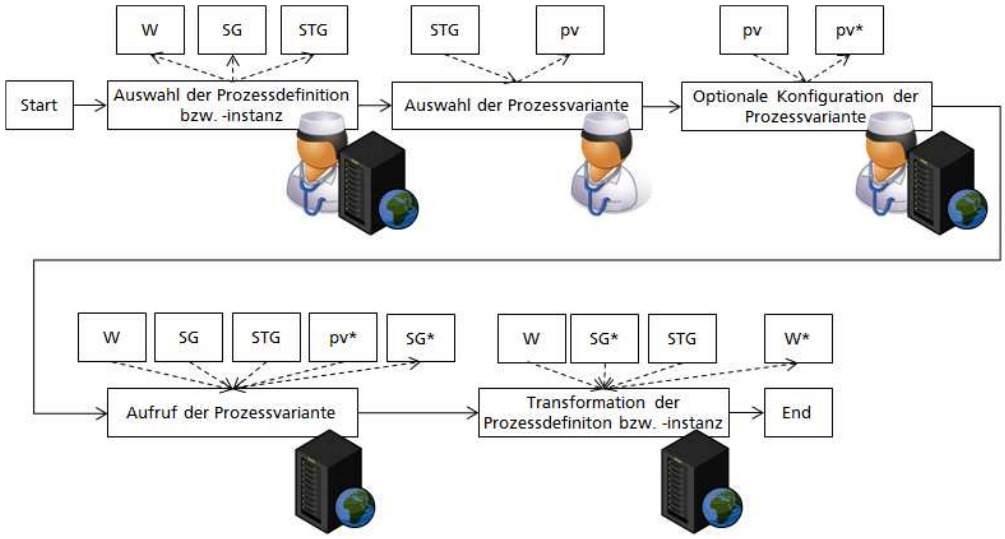


Die oben als Prozessdefinition dargestellte Prozessvariante besteht aus drei Änderungsoperationen:

– Einfügen und Konfiguration des Knoten *E* unterhalb von *B*
– Löschen des Knoten *D* unterhalb von *B* an erster Position
– Ersetzen des Knoten *E* unterhalb von *C* durch Knoten *F*

Wie aus der Abbildung deutlich wird, werden der Prozessvariante die Referenzen auf den SPF-Typ-Graphen (Datenobjekt »STG«), den SPF-Graphen (Datenobjekt »SG«) und die Prozessdefinition (Datenobjekt »W«) durch einen übergeordneten Prozess übergeben und sind daher direkt an den Startknoten der Variante geknüpft. Die Parameter über die hinzuzufügenden Knoten (Datenobjekte »E« und »F«), die Bedingungen für die Auswahl der Rekonfigurationsknoten bzw. der zu löschenden Knoten (Datenobjekte »Cond«) und die temporären Graphgrammatiken (Datenobjekte »temp-GG«) werden hingegen spezifisch für die Änderungsoperationen der Prozessvariante definiert. Das Datenobjekt »SG*« signalisiert die Verfügbarkeit einer neuen Version des SPF-Graphen *SG*. Die nachfolgende Abbildung zeigt, wie der übergeordnete Prozess, der so genannte »Top-Level Prozess« für den Aufruf von Prozessvarianten gestaltet werden kann.

Abbildung 118 Top-Level Prozess für den Aufruf von Prozessvarianten



Die Anwendung einer Prozessvariante auf einen SPF-Graphen erfolgt nicht vollkommen automatisiert, sondern erfordert Entscheidungen von Fachexperten. Zunächst muss die Auswahl der Prozessdefinition bzw. der Instanz getroffen werden, die modifiziert werden soll; diese Aktivität kann mit der Signalisierung des Bedarfs für eine Änderung und gegebenenfalls bereits mit der Auswahl einer Prozessvariante verbunden werden, da diese Auswahl immer nur ausgehend von einem bestimmten Prozess getroffen wird. Nach der Selektion der gewünschten Prozessvariante kann diese optional an die speziellen Anforderungen der Behandlungssituation angepasst werden; zu diesem Zweck wird sie analog zum SPF-Graphen des klinischen Pfades (re-)konfiguriert (siehe Kapitel 4.7.3). Anschließend wird die Variante für den mit dem Prozess verknüpften SPF-Graphen aufgerufen und führt zur Entstehung einer neuen Version dieses Graphen. Abschließend findet die Transformation der Prozessdefinition bzw. der Instanz ausgehend von der aktuellen Version des SPF-Graphen statt. Damit ist die Anwendung der Prozessvariante abgeschlossen.

4.7.3 Entwicklung konfigurierbarer Prozessvarianten auf Basis des Lösungskonzeptes

Die Erstellung von Prozessvarianten als Anwendungsprogramme oder Prozessdefinitionen kann vollkommen separat voneinander stattfinden. Ähnlich wie bei Prozessfragmenten, die in unterschiedlichen Prozessdefinitionen genutzt werden, können jedoch auch SPF-Änderungsoperationen oder Gruppen von Änderungsoperationen in unterschiedlichen Prozessvarianten eine Rolle spielen. In dem bereits beschriebenen Provop-Ansatz (siehe Kapitel 3.3.4) werden die bereitgestellten Änderungsoperationen daher zu so genannten Optionen gruppiert. In dem hier entwickelten Lösungskonzept sollen Prozessvarianten jedoch genau wie Prozessdefinitionen durch Transformation von SPF-Graphen hervorgehen. Je nach Bedarf kann also für eine Prozessvariante oder eine Menge von Prozessvarianten ein SPF-Typ-Graph definiert werden, aus dem dann durch Konfiguration die entsprechenden Varianten zunächst als SPF-Graphen abgeleitet werden.

Für eine Prozessvariante wird ein SPF-Typ-Graph definiert, wenn sich je nach Anwendungskontext für diese Variante unterschiedliche Ausprägungen ergeben können; z. B. gibt es für

eine Komplikation verschiedene Therapieverfahren, die in Abhängigkeit des Behandlungsfalles und der Wünsche des Patienten gewählt werden müssen. Dem gegenüber wird ein SPF-Typ-Graph über eine Gruppe von Prozessvarianten definiert, wenn die Ausgestaltung einer Variante von der Struktur anderer Varianten abhängt oder Varianten auf unterschiedliche Art und Weise miteinander kombiniert werden können; dies ist z. B. bei Multimorbidität relevant.

Ein SPF-Typ-Graph für Prozessvarianten entspricht dann einer Knotenhierarchie über SPF-Änderungsoperationen; d. h. jedes Prozessfragment dient der Umsetzung genau einer Änderungsoperation. Damit die Operationen erfolgreich ausgeführt werden können, müssen sie die Knotenbezeichnungen aus demselben SPF-Typ-Graphen referenzieren. Darüber hinaus muss der SPF-Graph, der manipuliert werden soll, ebenfalls von diesem SPF-Typ-Graphen abgeleitet worden sein. Obwohl SPF-Typ-Graphen für Prozessvarianten ebenso formal spezifiziert werden können, wie SPF-Typ-Graphen für Prozessfamilien (siehe Kapitel 4.3.5), müssen die Korrektheitskriterien aus Kapitel 4.3.6 erweitert werden. Sei $STG = (V, E, V_{SPF}, ...)$ ein SPF-Typ-Graph für Prozessvarianten, dann gelten zusätzlich die folgenden Kriterien:

Prozessfragmente als SPF-Änderungsoperationen. Bei den Prozessfragmenten, die über die Funktion *VP* des SPF-Typ-Graphen referenziert werden, darf es sich ausschließlich um SPF-Änderungsoperationen der Menge *O* handeln, wobei Elemente der Menge *O* ebenso wie Prozessfragmente der Menge *P* spezifiziert werden können. Sei $STG = (V, E, V_{SPF}, ...)$ ein SPF-Typ-Graph für Prozessvarianten, dann gilt:

$$VP(V_{SPF}) \subseteq O \subset P$$

Die folgende Tabelle zeigt ein Beispiel für die Spezifikation von Änderungsoperationen als Prozessfragmente ausgehend von den Festlegungen in Kapitel 4.3.4.

Tabelle 29 Beispiele für Schnittstellenspezifikationen von Prozessfragmenten als Bestandteile von Prozessvarianten

Identifikation Attribut	**1.0.23.7.111.201**	**1.0.23.7.111.202**
Name^{p}	Röntgenuntersuchungen durch MRT ersetzen	Blutzuckertest am Aufnahmetag einfügen
Description^{p}	Alle Röntgenuntersuchungen im Prozess werden durch MRT-Untersuchungen ersetzt.	Das Standardlabor am Aufnahmetag wird um die Durchführung eines Blutzuckertests erweitert
Version^{p}	1	1
Role^{p}	Case Manager	Case Manager
Method^{p}	1.0.23.7.331.900	1.0.23.7.652.21
Implementation^{p}	1.0.23.9.51.636	1.0.23.9.51.636

Da sämtliche Fragmente auf SPF-Änderungsoperationen basieren, kann die Implementierung bei allen Fragmenten dieselbe sein; z. B. handelt es sich um einen Web Service, der als Methoden die verschiedenen Änderungsoperationen bereitstellt. Die nächste Tabelle demonstriert die Spezifikation der Methode »Röntgenuntersuchungen durch MRT ersetzen« als SPF-Änderungsoperation INSERT.

Tabelle 30 Beispiel für die Spezifikation einer SPF-Änderungsoperation als Methode im Kontext von Prozessfragmenten

Identifikation Attribut	**1.0.23.7.331.900**
Name[m]	spfInsert
Description[m]	SPF-Änderungsoperation zum Einfügen und Konfigurieren nicht-terminaler Knoten in SPF-Graphen.
Parameters[m]	1.0.23.7.91.251, 1.0.23.7.201.17, 1.0.23.7.332.98, 1.0.23.7.332.99, 1.0.23.7.432.10, 1.0.23.7.86.41, 1.0.23.7.91.156

Einige der Parameter, die für den Aufruf der SPF-Änderungsoperation INSERT benötigt werden, können statisch definiert werden. Dies kann z. B. durch Angabe entsprechender Knotenattribute und fixen Wertzuweisungen im SPF-Typ-Graphen erfolgen. Eine andere Option besteht in der Erweiterung der Schnittstellenspezifikation von Parametern um ein Element *DataValue*, das den konkreten Wert des Datenobjekts umfasst. Bei der Implementierung muss dann eine geeignete Form für die Persistierung von SPF-Typ-Graphen und SPF-Graphen gewählt werden; je nach Art der Realisierung bietet sich hier die Nutzung von objektorientierten Datenbanken oder XML-Datenbanken an. Alternativ können sowohl statische als auch dynamische Datenobjekte über den Top-Level-Prozess angefordert und der Variante bei ihrer Instanziierung als Subprozess übergeben werden.

Tabelle 31 Spezifikation statischer Methodenparameter für die SPF-Änderungsoperation INSERT

Identifikation Attribut	**1.0.23.7.91.251**	**1.0.23.7.432.10**	**1.0.23.7.86.41**	**1.0.23.7.91.156**
Name[param]	STG	V^{STG}_{Add}	Conditions	temp-GG
Description[param]	SPF-Typ-Graph	Knoten MRT	(ALL, Röntgenuntersuchung, ∅ , ∅ ,∅ ,∅)	Temporäre Graphgrammatik
IOtype[param]	IN	IN	IN	IN
Demandmode[param]	MANDATORY	MANDATORY	MANDATORY	OPTIONAL
Datatype[param]	Userdefined: SPFTypeGraph	Userdefined: SPFGraph	Userdefined:Conditions	Userdefined: GraphGrammer

Der Parameter *Conditions* muss nicht komplett im Voraus festgelegt werden. Wie in Kapitel 4.7.1 bei der Spezifikation der SPF-Änderungsoperation INSERT gezeigt wurde, kann dies insbesondere bei der dynamischen Adaption von Prozessinstanzen problematisch sein. Aus diesem Grund können die Bedingungen für die Auswahl der Rekonfigurationsknoten alternativ auch dynamisch beeinflusst werden. Die Prozessvariante kann auf unterschiedliche SPF-Graphen angewandt werden, sofern diese von demselben SPF-Typ-Graphen abgeleitet wurden. Daher kann die Festlegung der Parameter *SG*, *W* und *I* nur dynamisch erfolgen.

Tabelle 32 Spezifikation dynamischer Methodenparameter für die SPF-Änderungsoperation INSERT

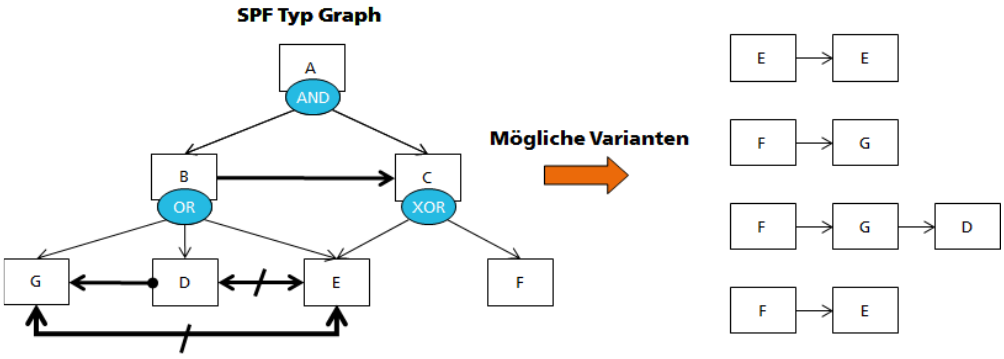
Identifikation Attribut	**1.0.23.7.201.17**	**1.0.23.7.332.98**	**1.0.23.7.332.99**
Name^{param}	SG	W	I
Description^{param}	SPF-Graph (dynamisch)	Prozessdefinition (dynamisch)	Prozessinstanz
IOtype^{param}	IN/OUT	IN	IN
Demandmode^{param}	MANDATORY	OPTIONAL	OPTIONAL
Datatype^{param}	Userdefined: SPF-Graph	Userdefined:ProcessDefinition	Userdefined: Process-Instance

Vollständige Sequenzialisierbarkeit. Bei der Transformation von SPF-Graphen auf Basis des SPF-Typ-Graphen für Prozessvarianten dürfen nur sequentielle Prozessdefinitionen entstehen, um zu vermeiden, dass SPF-Änderungsoperationen parallel zueinander ausgeführt werden. Dementsprechend müssen alle Nachfolger eines gemeinsamen Kontextknotens über Constraint-Relationen miteinander verknüpft sein:

$$\forall v_{Context} \in V \setminus V_{SPF}: VO(v_{Context}) \neq XOR \Rightarrow \forall v, u \in target(v_{Context}): v \neq u \wedge (\exists (v, u, ct \in ConstraintType) \in C \vee \exists (u, v, ct \in ConstraintType) \in C)$$

Diesem Kriterium zufolge müssen alle Knotenpaare unterhalb desselben Kontextknotens entweder über einen Anforderungs-, einen Sequenz- oder einen Ausschluss-Constraint miteinander verbunden sein. Dabei sind natürlich auch die Korrektheitskriterien aus Kapitel 4.3.6 zu beachten, die sich auf Konflikte zwischen verschiedenen Arten von Constraints oder problematische Kombinationen von logischen Operatoren und Constraints beziehen; so darf z. B. zwischen zwei Knoten unterhalb eines Kontextknotens mit AND-Operator kein Ausschluss-Constraint definiert werden. Auf Basis dieses Kriteriums ist es also möglich, die vollständige Sequenzialisierbarkeit der SPF-Änderungsoperationen nach der Transformation zu gewährleisten. Die nachfolgende Abbildung demonstriert an einem Beispiel die möglichen sequentiellen Ausführungsreihenfolgen.

Abbildung 119 Auswirkung des Korrektheitskriteriums für vollständige Sequenzialisierbarkeit auf die Struktur von SPF-Typ-Graphen



Wie das obige Beispiel zeigt, sind nur sequentielle Strukturen für Prozessvarianten möglich, sofern das Kriterium für die vollständige Sequenzialisierbarkeit bei der Erstellung des SPF-Typ-Graphen beachtet wird. Konfiguration und Transformation folgen exakt dem in den Kapiteln 4.4 und 4.5 beschriebenen Vorgehen.

Auf Basis der hier vorgestellten Methodik lassen sich Prozessvarianten auf dieselbe Art und Weise wie klinische Pfade erstellen: Die formale Grundlage bildet bei beiden Prozesstypen das in dieser Arbeit entwickelte Lösungskonzept. Während der Definitionsphase wird ein SPF-Typ-Graph entweder für genau eine Prozessvariante oder eine Menge von Varianten definiert, die in gegenseitiger Wechselbeziehung stehen. Die Implementierung sämtlicher Prozessfragmente basiert auf Methodenaufrufen von SPF-Änderungsoperationen. In der Konfigurationsphase werden aus dem SPF-Typ-Graphen konkrete Prozessvarianten abgeleitet, die dem Standardvorgehen bei Auftreten einer bestimmten Prozessabweichung entsprechen. Durch Transformation werden die deklarativen SPF-Graphen in ausführbare Prozessdefinitionen umgewandelt. Diese können nun instanziiert und ausgeführt werden, um die Prozessdefinition oder –instanz eines klinischen Pfades ad hoc zu modifizieren. Mit der Rekonfigurationsphase besteht darüber hinaus auch die Möglichkeit, die Struktur einer Prozessvariante flexibel zu ändern, wenn der durch sie vorgegebene »best practice« Ansatz für den aktuellen Behandlungsfall nicht geeignet ist.

4.7.4 Zusammenfassung der Ergebnisse

In diesem Kapitel wurde gezeigt, wie Prozessvarianten realisiert werden können, um komplexe Änderungen an Prozessdefinitionen und ihren Instanzen zu erreichen. Der Aufwand für Fachexperten zur Modifikation des klinischen Pfades aufgrund einer Vorerkrankung, einer Komplikation oder organisatorischen Änderungen reduziert sich damit von vielen manuellen Rekonfigurationsschritten auf die Auswahl und Anwendung einer Prozessvariante. Jede Variante setzt sich aus Einzelmodifikationen, so genannten SPF-Änderungsoperationen zusammen. Eine SPF-Änderungsoperation dient der Manipulation eines SPF-Graphen, z. B. durch das Einfügen und Konfigurieren nicht-terminaler Knoten. Die Identifikation der Positionen im SPF-Graphen, an dem neue Knoten eingefügt werden sollen, oder die Ermittlung von Knoten, die gelöscht oder rekonfiguriert werden sollen, basiert auf dem Algorithmus zum Auffinden von Knoten, der in Kapitel 4.6.3 vorgestellt worden ist.

Auf die Spezifikation der SPF-Änderungsoperationen folgt die Definition von Prozessvarianten als geordnete Menge von solchen Operationen. An Hand eines Top-Level Prozesses wurde gezeigt, welche Schritte notwendig sind, um eine Variante aufzurufen. Ähnlich wie klinische Pfade müssen auch Prozessvarianten bei Bedarf konfiguriert werden können, um individuellen Erfordernissen eines Behandlungsfalles zu genügen. Gerade bei multimorbiden Patienten, die z. B. unter Bluthochdruck, Diabetes mellitus und Adipositas leiden, ist es notwendig, Prozessvarianten miteinander zu kombinieren. Die Auswahl einer oder mehrerer standardisierter Varianten reicht nicht immer aus, um den speziellen Bedürfnissen solcher Patienten gerecht zu werden. Aus diesem Grund wurde abschließend ein Verfahren beschrieben, wie das Lösungskonzept zur Abbildung, Konfiguration, Transformation und Rekonfiguration klinischer Pfade auch auf Prozessvarianten übertragen werden kann. Dabei wurden die Korrektheitskriterien, die allgemein für SPF-Typ-Graphen aufgestellt worden sind, noch um Aspekte erweitert, die speziell dazu dienen, SPF-Typ-Graphen für Prozessvarianten zu entwickeln. Zusammenfassend wurden die folgenden Ergebnisse erarbeitet:

- Spezifikation von SPF-Änderungsoperationen zur Manipulation von SPF-Graphen durch Einfügen, Löschen, Ersetzen und Rekonfigurieren von Knoten
- Spezifikation von Prozessvarianten als geordnete Menge von SPF-Änderungsoperationen, die in sequentieller Reihenfolge ausgeführt werden
- Entwicklung eines Konzepts zur Umsetzung konfigurierbarer Prozessvarianten und Erweiterung der Korrektheitskriterien für SPF-Typ-Graphen

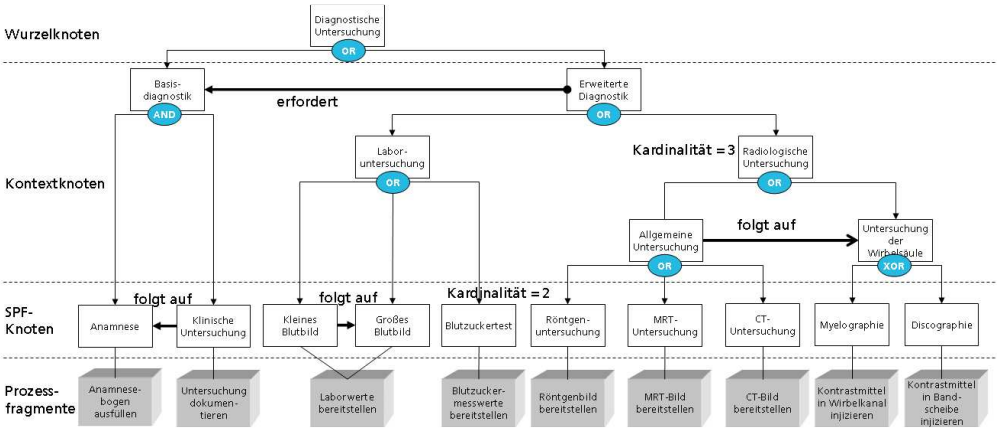
5 Validierung des Lösungskonzepts an Hand einer Fallstudie

In Kapitel 4 wurde das gesamte Lösungskonzept zur Entwicklung klinischer Pfade bestehend aus den Phasen Definition, Konfiguration, Transformation und Rekonfiguration vorgestellt sowie ein Verfahren zur Realisierung von Prozessvarianten entwickelt. Das Lösungskonzept soll nun an Hand einer Fallstudie validiert werden, wobei die Umsetzung der einzelnen Phasen und ihr Ineinandergreifen beschrieben werden. Die Inhalte der Fallstudie wurden bereits in Kapitel 2.2.1 präsentiert; es handelt sich um die stationäre Behandlung von Wirbelsäulenerkrankungen.

5.1 Definitionsphase

In der Definitionsphase wird ein SPF-Typ-Graph erstellt, der sämtliche Aspekte der Behandlung von Wirbelsäulenerkrankungen abdecken soll. Bei Bedarf kann dieser Graph später sukzessive um weitere Diagnosen und Therapieformen angereichert werden. Aus Gründen der Übersichtlichkeit können nicht alle Aspekte des Behandlungsszenarios dargestellt werden. Zur Verbesserung der Darstellbarkeit wurde der umfassende SPF-Typ-Graph, der hier vorgestellt werden soll, in zwei Graphen zur Repräsentation der Behandlungsabschnitte Diagnostik bzw. Therapie unterteilt. Die XML-Repräsentationen der SPF-Typ-Graphen für Diagnostik finden sich in Anhang A.1.

Abbildung 120 SPF-Typ-Graph für die Diagnostik von Wirbelsäulenerkrankungen



Diesem SPF-Typ-Graphen zu Folge besteht jede diagnostische Untersuchung mindestens aus der Basisdiagnostik oder – aufgrund des Anforderungs-Constraints – aus der Basisdiagnostik und der erweiterten Diagnostik. Die Basisdiagnostik wiederum setzt sich aus der Anamnese und der klinischen Untersuchung zusammen. Enthält eine Prozessdefinition beide Prozessfragmente, so wird das Fragment »Anamnesebogen ausfüllen« wegen des Sequenz-Constraints stets vor dem Fragment »Untersuchung dokumentieren« ausgeführt. Die erweitere Diagnostik umfasst wenigstens eine Laboruntersuchung oder eine radiologische Untersuchung. Bei der Laboruntersuchung werden kleines und großes Blutbild voneinander unterschieden; beide können jedoch auf dasselbe Prozessfragment abgebildet werden. In der Pro-

zessdefinition unterscheiden sie sich dann nur noch durch die Knotenidentität aus dem SPF-Graphen und ihre Attribute voneinander. Darüber hinaus besteht auch die Möglichkeit der Durchführung von maximal zwei Blutzuckertests. Bei der radiologischen Untersuchung wird differenziert nach allgemeinen Untersuchungen und speziellen Untersuchungen der Wirbelsäule. Bei den allgemeinen Untersuchungen müssen sich Fachexperten für mindestens eines der Bild-gebenden Verfahren Röntgen, MRT und CT entscheiden. Bei den Wirbelsäulenbezogenen Untersuchungen handelt es sich um Myelographie und Discographie. Bei der Myelographie wird ein Konstrastmittel direkt in den Wirbelkanal injiziert, bevor anschließend Röntgen- oder CT-Bilder angefertigt werden. Aus der Verteilung des Kontrastmittels kann anschließend geschlossen werden, ob Nerven oder Rückenmark im Wirbelkanal gedrückt werden. Die Myelographie ist außerdem nützlich, um einen wiederkehrenden Bandscheibenvorfall (Rezidiv) von postoperativen Vernarbungen zu unterscheiden. Bei der Discographie wird das Kontrastmittel in die Bandscheibe gespritzt; diese Methode dient der Unterscheidung zwischen discogenen (d.h. Bandscheiben-bedingten) Schmerzen und dem Facetten-Syndrom. In einem anschließend erstellten CT kann beurteilt werden, welche Anteile der Bandscheibe geschädigt sind. Insgesamt können bis zu drei radiologische Untersuchungen in einem Diagnostikprozess durchgeführt werden.

An diesem Beispiel lässt sich erkennen, dass für das Anlegen des SPF-Typ-Graphen grundlegende Kenntnisse über Zusammenhänge des abzubildenden Behandlungsprozesses ausreichen. Z. B. muss der IT-Spezialist, dessen Aufgabe die Entwicklung des Graphen ist, nicht unbedingt wissen, dass nach der Injektion von Kontrastmittel entweder ein Röntgenbild oder ein CT erstellt werden, aber kein MRT. Da Fachexperten, also Ärztinnen und Ärzte, die Konfiguration des SPF-Typ-Graphen und damit die Komposition der Prozessfragmente übernehmen, liegt es in ihrer Verantwortung, semantisch korrekte Prozessdefinitionen zu generieren. Soll dennoch gänzlich vermieden werden, dass nach einer Myelographie oder einer Discographie ein MRT gewählt werden kann, so kann der SPF-Typ-Graph auch entsprechend umstrukturiert werden. Durch die kompetenzorientierte Aufgabenteilung zwischen Domänenexperten und IT-Spezialisten lassen sich solche Fehlerquellen jedoch in der Regel auch ohne detaillierte Spezifikationen vermeiden.

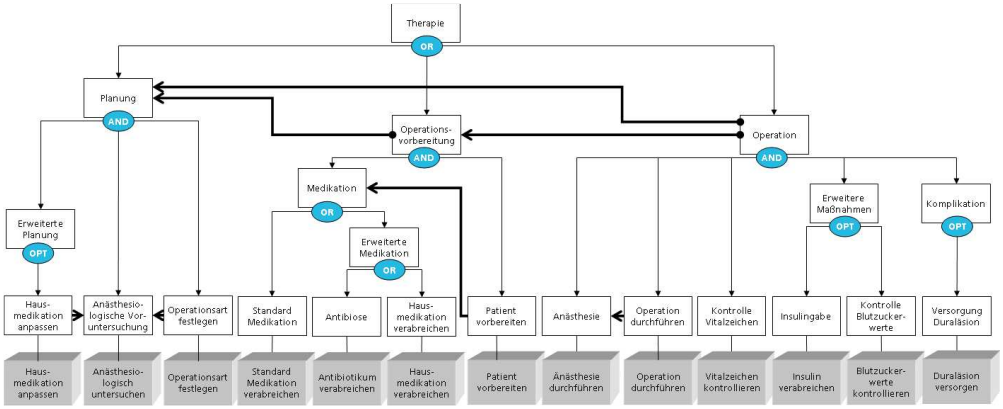
Für jedes Prozessfragment müssen die IT-Spezialisten eine Schnittstellenspezifikation gemäß den Beschreibungen in Kapitel 4.3.4 erstellen. Diese Spezifikation umfasst neben Metadaten wie Name und Beschreibung auch einen Verweis auf die jeweilige Implementierung, den Methodenaufruf und die notwendigen Parameter. Die folgende Tabelle zeigt beispielhaft die Schnittstellenspezifikation für das Prozessfragment »Anamnesebogen ausfüllen«; dabei kann es sich z. B. um ein Online-Formular oder ein MS Word-Formular handeln, das in der elektronischen Patientenakte im Krankenhausinformationssystem abgelegt wird.

Tabelle 33 Schnittstellenspezifikation für das Prozessfragment »Anamnesebogen ausfüllen«

Identifikation Attribut	**1.0.23.7.871.97**
Name[p]	Anamnesebogen ausfüllen
Description[p]	Anamnesedaten eintragen und in Patientenakte einstellen
Version[p]	2
Role[p]	Aufnahmearzt
Method[p]	1.0.23.7.441.21
Implementation[p]	1.0.23.9.52.624

Für die Spezifikation der Methoden, Aufrufparameter und der übrigen Prozessfragmente sei an dieser Stelle auf Beispiele aus Kapitel 4.3.4 verwiesen. Die nächste Abbildung zeigt den SPF-Typ-Graph für die Therapie von Wirbelsäulenerkrankungen.

Abbildung 121 SPF-Typ-Graph für die Therapie von Wirbelsäulenerkrankungen



Ähnlich wie der SPF-Typ-Graph für die Diagnostik lässt sich auch der Graph für die Therapie von Wirbelsäulenerkrankungen weit detailreicher darstellen, als oben abgebildet. Dennoch vermittelt dieser vereinfachte Graph einen guten Überblick über die wesentlichen Therapieelemente. Jede Therapie beginnt mit der Planung; bei Bedarf kann die Behandlung schon nach Abschluss der Therapieplanung abgebrochen werden, z. B. wenn der Patient aufgrund seines allgemeinen Gesundheitszustands nicht operiert werden kann. Im Normalfall folgen der Planung jedoch die Operationsvorbereitung und schließlich die Operation. Zur Therapieplanung gehören mindestens die anästhesiologische Voruntersuchung und die Festlegung der konkreten Operationsart, wie z. B. die mikrochirurgische Bandscheiben-OP, das Einsetzen einer künstlichen Bandscheibe oder die Wirbelsäulenversteifung. Darüber hinaus sind in Abhängigkeit eines konkreten Behandlungsfalles weitere Maßnahmen erforderlich; bei chronisch kranken Patienten betrifft dies insbesondere die Anpassung der häuslichen Medikation. Die Operationsvorbereitung beinhaltet die Medikation des Patienten und seine konkrete Vorbereitung für die Operation (z. B. Transport des Patienten in den OP-Raum und Positionierung auf dem OP-Tisch). Neben der Standardmedikation, zu der beispielsweise Beruhigungspräparate zählen, kann eine spezifische Medikation erforderlich sein, die sich auf die Hausapotheke oder auch eine Antibiose bezieht; eine Antibiose ist notwendig, wenn beim Patienten ein erhöhtes Infektionsrisiko besteht. Im Anschluss an die Vorbereitung erfolgt die eigentliche Operation. Bereits während der Anästhesie und im Laufe der gesamten Operation werden die Vitalzeichen des Patienten überwacht. Erweiterte Maßnahmen für Diabetiker beziehen sich auf die intravenöse Gabe von Insulin und die Kontrolle der Blutzuckerwerte. Einige Komplikationen,

wie z. B. die Duraläsion, können intraoperativ behandelt werden; bei der Duraläsion handelt es sich um eine Verletzung an der Haut des Wirbelkanals.

5.2 Konfigurationsphase

In der Konfigurationsphase wird aus dem SPF-Typ-Graphen für die Diagnostik von Wirbelsäulenerkrankungen ein konkreter klinischer Pfad als Prozessstandard in Form eines SPF-Graphen abgeleitet. Zu diesem Zweck wird aus dem SPF-Typ-Graphen automatisch eine Graphgrammatik *GG = (STG, PR, START)* generiert. Für den SPF-Typ-Graphen zur Diagnostik von Wirbelsäulenerkrankungen umfasst die Menge *PR* die folgenden Produktionsregeln:

Formel 29 Graphgrammatik für den SPF-Graph zur Diagnostik von Wirbelsäulenerkrankungen

$$
\begin{aligned}
N_{\text{Diagnostische Untersuchung}} &\rightarrow N^{1}_{\text{Diagnostische Untersuchung}}\\
N^{1}_{\text{Diagnostische Untersuchung}} &\rightarrow \text{Diagnostische Untersuchung } (N_{\text{Basisdiagnostik}})\\
N^{1}_{\text{Diagnostische Untersuchung}} &\rightarrow \text{Diagnostische Untersuchung } (N_{\text{Erweiterte Diagnostik}})\\
N^{1}_{\text{Diagnostische Untersuchung}} &\rightarrow \text{Diagnostische Untersuchung } (N_{\text{Basisdiagnostik}}, N_{\text{Erweiterte Diagnostik}})\\
\\
N_{\text{Basisdiagnostik}} &\rightarrow N^{1}_{\text{Basisdiagnostik}}\\
N^{1}_{\text{Basisdiagnostik}} &\rightarrow \text{Basisdiagnostik } (N_{\text{Anamnese}}, N_{\text{Klinische Untersuchung}})\\
\\
N_{\text{Anamnese}} &\rightarrow N^{1}_{\text{Anamnese}}\\
N^{1}_{\text{Anamnese}} &\rightarrow \text{Anamnese}\\
\\
N_{\text{Klinische Untersuchung}} &\rightarrow N^{1}_{\text{Klinische Untersuchung}}\\
N^{1}_{\text{Klinische Untersuchung}} &\rightarrow \text{Klinische Untersuchung}\\
\\
N_{\text{Erweiterte Diagnostik}} &\rightarrow N^{1}_{\text{Erweiterte Diagnostik}}\\
N^{1}_{\text{Erweiterte Diagnostik}} &\rightarrow \text{Erweiterte Diagnostik } (N_{\text{Laboruntersuchung}})\\
N^{1}_{\text{Erweiterte Diagnostik}} &\rightarrow \text{Erweiterte Diagnostik } (N_{\text{Radiologische Untersuchung}})\\
N^{1}_{\text{Erweiterte Diagnostik}} &\rightarrow \text{Erweiterte Diagnostik } (N_{\text{Laboruntersuchung}}, N_{\text{Radiologische Untersuchung}})\\
\\
N_{\text{Laboruntersuchung}} &\rightarrow N^{1}_{\text{Laboruntersuchung}}\\
N^{1}_{\text{Laboruntersuchung}} &\rightarrow \text{Laboruntersuchung } (N_{\text{Kleines Blutbild}})\\
N^{1}_{\text{Laboruntersuchung}} &\rightarrow \text{Laboruntersuchung } (N_{\text{Großes Blutbild}})\\
N^{1}_{\text{Laboruntersuchung}} &\rightarrow \text{Laboruntersuchung } (N_{\text{Blutzuckertest}})\\
N^{1}_{\text{Laboruntersuchung}} &\rightarrow \text{Laboruntersuchung } (N_{\text{Kleines Blutbild}}, N_{\text{Großes Blutbild}})\\
N^{1}_{\text{Laboruntersuchung}} &\rightarrow \text{Laboruntersuchung } (N_{\text{Kleines Blutbild}}, N_{\text{Blutzuckertest}})\\
N^{1}_{\text{Laboruntersuchung}} &\rightarrow \text{Laboruntersuchung } (N_{\text{Großes Blutbild}}, N_{\text{Blutzuckertest}})\\
N^{1}_{\text{Laboruntersuchung}} &\rightarrow \text{Laboruntersuchung } (N_{\text{Kleines Blutbild}}, N_{\text{Großes Blutbild}}, N_{\text{Blutzuckertest}})\\
\\
N_{\text{Kleines Blutbild}} &\rightarrow N^{1}_{\text{Kleines Blutbild}}\\
N^{1}_{\text{Kleines Blutbild}} &\rightarrow \text{Kleines Blutbild}\\
\\
N_{\text{Großes Blutbild}} &\rightarrow N^{1}_{\text{Großes Blutbild}}\\
N^{1}_{\text{Großes Blutbild}} &\rightarrow \text{Großes Blutbild}\\
\\
N_{\text{Blutzuckertest}} &\rightarrow N^{1}_{\text{Blutzuckertest}}\\
N^{1}_{\text{Blutzuckertest}} &\rightarrow \text{Blutzuckertest}\\
\\
N_{\text{Radiologische Untersuchung}} &\rightarrow N^{1}_{\text{Radiologische Untersuchung}}\\
N_{\text{Radiologische Untersuchung}} &\rightarrow N^{2}_{\text{Radiologische Untersuchung}}\\
N_{\text{Radiologische Untersuchung}} &\rightarrow N^{3}_{\text{Radiologische Untersuchung}}\\
N^{2}_{\text{Radiologische Untersuchung}} &\rightarrow N^{1}_{\text{Radiologische Untersuchung}}, N^{1}_{\text{Radiologische Untersuchung}}\\
N^{3}_{\text{Radiologische Untersuchung}} &\rightarrow N^{1}_{\text{Radiologische Untersuchung}}, N^{2}_{\text{Radiologische Untersuchung}}
\end{aligned}
$$

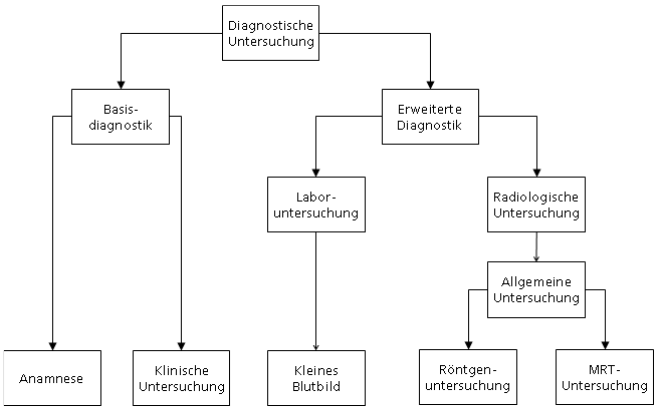
$N^1_{\text{Radiologische Untersuchung}}$	→	Radiologische Untersuchung ($N_{\text{Allgemeine Untersuchung}}$)
$N^1_{\text{Radiologische Untersuchung}}$	→	Radiologische Untersuchung ($N_{\text{Untersuchung der Wirbelsäule}}$)
$N^1_{\text{Radiologische Untersuchung}}$	→	Radiologische Untersuchung ($N_{\text{Allgemeine Untersuchung}}$, $N_{\text{Untersuchung der Wirbelsäule}}$)
$N_{\text{Allgemeine Untersuchung}}$	→	$N^1_{\text{Allgemeine Untersuchung}}$
$N^1_{\text{Allgemeine Untersuchung}}$	→	Allgemeine Untersuchung ($N_{\text{Röntgenuntersuchung}}$)
$N^1_{\text{Allgemeine Untersuchung}}$	→	Allgemeine Untersuchung ($N_{\text{MRT-Untersuchung}}$)
$N^1_{\text{Allgemeine Untersuchung}}$	→	Allgemeine Untersuchung ($N_{\text{CT-Untersuchung}}$)
$N^1_{\text{Allgemeine Untersuchung}}$	→	Allgemeine Untersuchung ($N_{\text{Röntgenuntersuchung}}$, $N_{\text{MRT-Untersuchung}}$)
$N^1_{\text{Allgemeine Untersuchung}}$	→	Allgemeine Untersuchung ($N_{\text{Röntgenuntersuchung}}$, $N_{\text{CT-Untersuchung}}$)
$N^1_{\text{Allgemeine Untersuchung}}$	→	Allgemeine Untersuchung ($N_{\text{MRT-Untersuchung}}$, $N_{\text{CT-Untersuchung}}$)
$N^1_{\text{Allgemeine Untersuchung}}$	→	Allgemeine Untersuchung ($N_{\text{Röntgenuntersuchung}}$, $N_{\text{MRT-Untersuchung}}$, $N_{\text{CT-Untersuchung}}$)
$N_{\text{Röntgenuntersuchung}}$	→	$N^1_{\text{Röntgenuntersuchung}}$
$N^1_{\text{Röntgenuntersuchung}}$	→	Röntgenuntersuchung
$N_{\text{MRT-Untersuchung}}$	→	$N^1_{\text{MRT-Untersuchung}}$
$N^1_{\text{MRT-Untersuchung}}$	→	MRT-Untersuchung
$N_{\text{CT-Untersuchung}}$	→	$N^1_{\text{CT-Untersuchung}}$
$N^1_{\text{CT-Untersuchung}}$	→	CT-Untersuchung
$N_{\text{Untersuchung der Wirbelsäule}}$	→	$N^1_{\text{Untersuchung der Wirbelsäule}}$
$N^1_{\text{Untersuchung der Wirbelsäule}}$	→	Untersuchung der Wirbelsäule ($N_{\text{Myelographie}}$)
$N^1_{\text{Untersuchung der Wirbelsäule}}$	→	Untersuchung der Wirbelsäule ($N_{\text{Discographie}}$)
$N_{\text{Myelographie}}$	→	$N^1_{\text{Myelographie}}$
$N^1_{\text{Myelographie}}$	→	Myelographie
$N_{\text{Discographie}}$	→	$N^1_{\text{Discographie}}$
$N^1_{\text{Discographie}}$	→	Discographie

Das Startsymbol der Graphgrammatik wird auf einen SPF-Graphen gesetzt, der lediglich aus dem Knoten »Diagnostische Untersuchung« besteht:

$START = (N_{\text{Diagnostische Untersuchung}}, \varnothing, \varnothing, N_{\text{Diagnostische Untersuchung}}, N_{\text{Diagnostische Untersuchung}}, A, VA, VID)$

Durch Anwendung der Regeln dieser Graphgrammatik kann nun der klinische Pfad aus dem allgemeinen Prozesswissen erstellt werden. Die nächste Abbildung zeigt den SPF-Graphen für den klinischen Pfad zur Diagnostik von Wirbelsäulenerkrankungen als Ergebnis der Konfiguration.

Abbildung 122 SPF-Graph für den klinischen Pfad zur Diagnostik von Wirbelsäulenerkrankungen

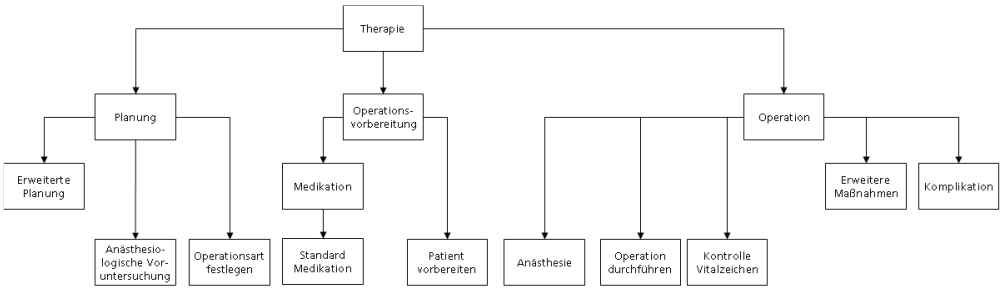


Der klinische Pfad umfasst also die Basisdiagnostik, die Anfertigung eines kleinen Blutbilds sowie einer Röntgen- und einer MRT-Untersuchung. Gemäß den Regeln der Graphgrammatik wird der klinische Pfad durch folgenden Ausdruck repräsentiert:

(Diagnostische Untersuchung
(Basisdiagnostik
(Anamnese, Klinische Untersuchung),
Erweiterte Diagnostik
(Laboruntersuchung
(Kleines Blutbild),
Radiologische Untersuchung
(Allgemeine Untersuchung
(Röntgenuntersuchung, MRT-Untersuchung)))))

Die Graphgrammatik für die Therapie von Wirbelsäulenerkrankungen wird auf analoge Weise generiert. Die nachstehende Abbildung zeigt den SPF-Typ-Graphen für den entsprechenden klinischen Pfad.

Abbildung 123 SPF-Graph für den klinischen Pfad zur Therapie von Wirbelsäulenerkrankungen



Der klinische Pfad umfasst sowohl Planung als auch Operationsvorbereitung und Operation. Nach der anästhesiologischen Voruntersuchung und der Festlegung der Operationsart erfolgt die Vorbereitung des Patienten auf die OP. Verläuft die Operation komplikationsfrei, besteht der SPF-Graph hier lediglich aus den Knoten »Anästhesie«, »Operation durchführen« und »Kontrolle Vitalzeichen«.

5.3 Transformationsphase

Während der Transformationsphase sollen die SPF-Graphen zur Diagnostik und Therapie von Wirbelsäulenerkrankungen nun in prozedurale Prozessdefinitionen umgewandelt werden, die für Patienten instanziiert und ausgeführt werden können. Diese Transformation wird nun detailliert am Beispiel des SPF-Graphen zur Diagnostik beschrieben. Zuerst wird die Menge der transformierenden Knoten V_{Trans} im SPF-Graphen ermittelt; dabei handelt es sich um alle SPF-Knoten, also Knoten, denen im SPF-Typ-Graphen Prozessfragmente zugeordnet wurden.

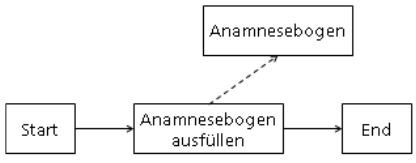
V_{Trans} = *{Anamnese, Klinische Untersuchung, Kleines Blutbild, Röntgenuntersuchung, MRT-Untersuchung}*

Im Anschluss erfolgt die Komposition der Prozessfragmente zu einer Prozessdefinition als ADEPT2 WSM Net. Für jeden Knoten wird die Menge N_{Before} derjenigen Aktivitäten in der Prozessdefinition bestimmt, die vorher ausgeführt werden müssen; diese Menge umfasst auch sämtliche Klonaktivitäten. Analog wird auch die Menge N_{After} ermittelt, die alle Prozessaktivitäten enthält, die erst nach dem aktuell zu transformierenden Prozessfragment bearbeitet werden sollen. Die Reihenfolge, in der die Aktivitäten in der Prozessdefinition angeordnet werden, wird also durch die Constraint-Relationen determiniert. Zugunsten der Lesbarkeit werden die Knotenidentitäten in den Namen der Prozessaktivitäten nicht dargestellt; d. h. der Name der Aktivität entspricht hier lediglich dem Namen des Prozessfragments. Welche Datenobjekte von einem Prozessfragment gelesen und geschrieben werden, ist in der Schnittstellenspezifikation des jeweiligen Fragments angegeben. Sei nun *W* die initial erzeugte Prozessdefinition, bestehend aus einem Start- und einem Endknoten, die über eine gerichtete Kontrollflusskante miteinander verbunden sind.

Knoten »Anamnese« transformieren:

N_{Before} = ∅
N_{After} = ∅
serialInsert(W, »Start«, »End«, »Anamnese«, Implementation^{p1})
addDataElements(W, »Anamnesebogen«, »Userdefined:Word«, »NULL«)
addDataEdges(W, »Anamnese«, »Anamnesebogen«, write)

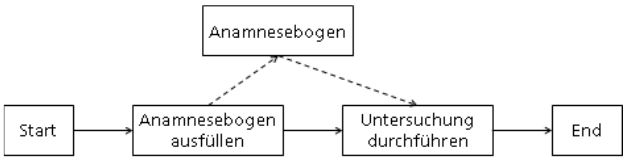
Abbildung 124 Transformation des »Knoten Anamnese«



Knoten »Klinische Untersuchung« transformieren:

N_{Before} = *{Anamnesebogen ausfüllen}*
N_{After} = ∅
serialInsert(W, »Anamnesebogen ausfüllen«, »End«, »Untersuchung durchführen«, Implementation^{p2})
addDataEdges(W, »Untersuchung durchführen«, »Anamnesebogen«, read)

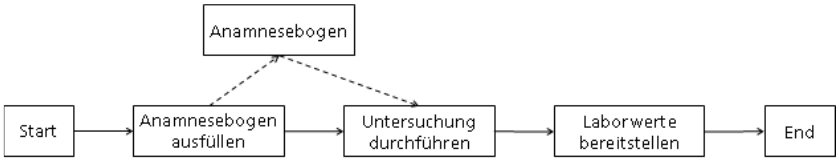
Abbildung 125 Transformation des Knoten »Klinische Untersuchung«



Knoten »Kleines Blutbild« transformieren:

N_{Before} = {Anamnesebogen ausfüllen, Untersuchung durchführen}
N_{After} = ∅
serialInsert(W, »Untersuchung durchführen«, »End«, »Laborwerte bereitstellen«, Implementation^{p3})

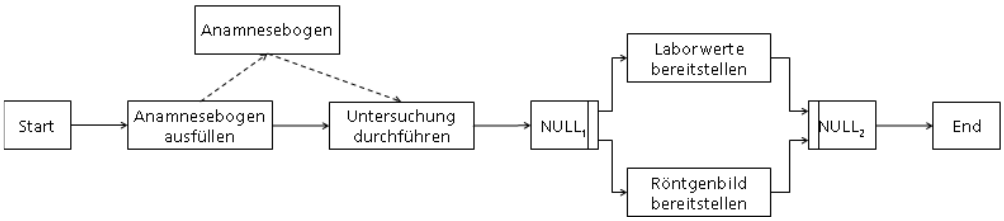
Abbildung 126 Transformation des Knoten »Kleines Blutbild«



Knoten »Röntgenuntersuchung« transformieren:

N_{Before} = {Anamnesebogen ausfüllen, Untersuchung durchführen}
N_{After} = ∅
parallelInsert(W, »Laborwerte bereitstellen«, »Laborwerte bereitstellen«, »Röntgenbild bereitstellen«, Implementation^{p4})

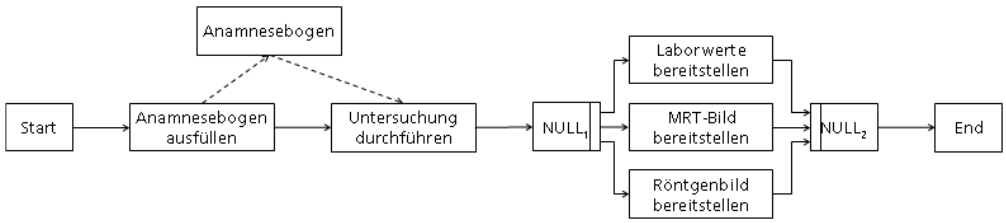
Abbildung 127 Transformation des Knoten »Röntgenuntersuchung«



Knoten »MRT-Untersuchung« transformieren:

N_{Before} = {Anamnesebogen ausfüllen, Untersuchung durchführen}
N_{After} = ∅
parallelInsert(W, »$NULL_1$«, »$NULL_2$«, »MRT-Bild bereitstellen«, Implementation^{p5})

Abbildung 128 Transformation des Knoten »MRT-Untersuchung«

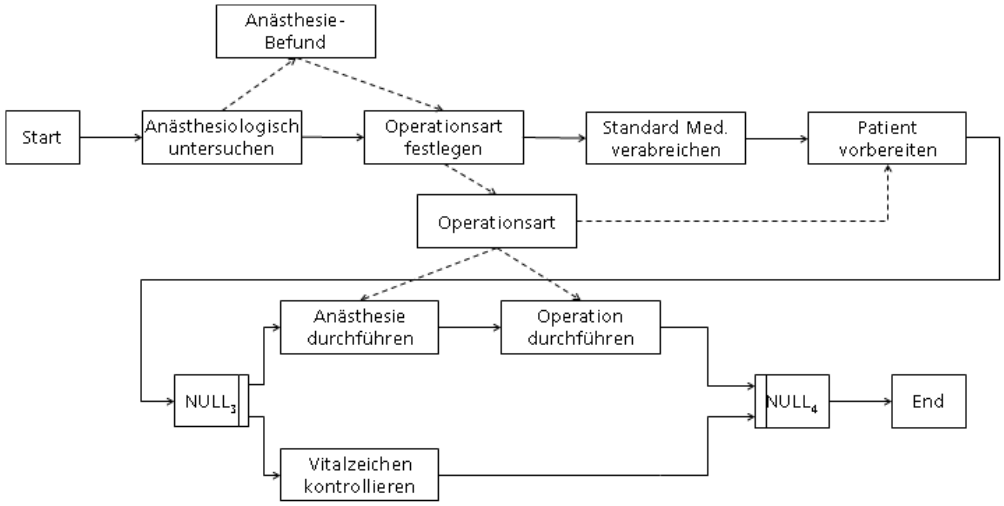


Analog zum klinischen Pfad zur Diagnostik von Wirbelsäulenerkrankungen lässt, sich auch der Pfad zur Therapie transformieren.

> V_{Trans} = *{Anästhesiologische Voruntersuchung, Operationsart festlegen, Standard Medikation, Patient vorbereiten, Anästhesie, Operation durchführen, Kontrolle Vitalzeichen}*

Die nachfolgende Abbildung zeigt den Therapieprozess als WSM Net nach der Transformation des SPF-Graphen.

Abbildung 129 Prozessdefinition für die Therapie von Wirbelsäulenerkrankungen nach der Transformation des SPF-Graphen



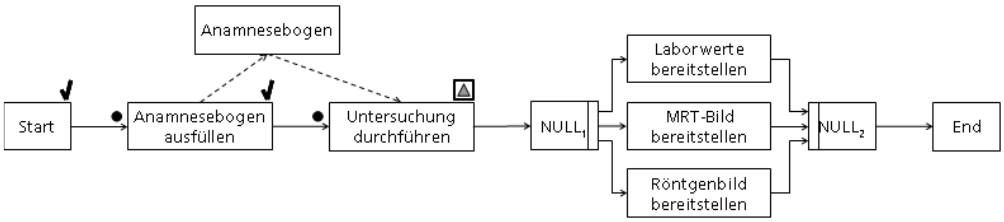
Nach der Transformation sind beide Prozessdefinitionen auf Grundlage der Prozessfragmente vollständig operationalisiert und direkt ausführbar.

5.4 Rekonfigurationsphase

Die Rekonfigurationsphase ermöglicht beim Auftreten einer Varianz vom klinischen Pfad die Umstrukturierung von Prozessdefinitionen und –instanzen durch das Hinzufügen neuer Knoten, das Löschen von Knoten oder die Rekonfiguration von Knoten im SPF-Graphen. An Hand des vom Patienten ausgefüllten Anamnesebogens stellt der Aufnahmearzt fest, dass der Patient, für den der klinische Pfad zur Diagnostik von Wirbelsäulenerkrankungen ausgeführt wird, schon einmal an der Wirbelsäule operiert worden ist. Daher muss nun abgeklärt werden, ob wirklich ein rezidiver Bandscheibenvorfall vorliegt oder ob es sich lediglich um vernarbtes Ge-

webe handelt, das Probleme verursacht. Die dafür benötigten Befunde liegen zu dem Zeitpunkt der Aufnahme des Patienten im Krankenhaus noch nicht vor (siehe Kapitel 2.2.1.5). Statt der standardmäßig im klinischen Pfad vorgesehenen Röntgen- und MRT-Untersuchung soll nun eine Myelographie durchgeführt werde. Die nächste Abbildung illustriert die aktuelle Prozessinstanz.

Abbildung 130 Instanz des klinischen Pfades zur Diagnostik von Wirbelsäulenerkrankungen



Um die Prozessinstanz ad hoc zu verändern, wird zunächst der Knoten »MRT-Untersuchung« unterhalb des Kontextknotens »Allgemeine Untersuchung« strukturerhaltend aus dem SPF-Graphen entfernt. Anschließend muss der Kontextknoten »Radiologische Untersuchung« im SPF-Graphen rekonfiguriert werden. Durch eine strukturerhaltende Rekonfiguration wird der nicht-terminale Knoten für »Untersuchung der Wirbelsäule« in den SPF-Graphen eingefügt. Dieser wird anschließend vom Arzt so konfiguriert, dass er als Nachfolger den Knoten »Myelographie« enthält. Aus Sicht des Fachexperten ähnelt die Rekonfiguration weitgehend dem Vorgehen bei der Konfiguration. Sei *STG* der SPF-Typ-Graph und SG^I der SPF-Graph des klinischen Pfades für die Instanz *I*:

deleteNodeRuntime(STG, SG^I, I, »Allgemeine Untersuchung«, »MRT-Untersuchung«)
addNodeRuntime(STG, SG^I, I, »Radiologische Untersuchung«, »$N_{Untersuchung\ der\ Wirbelsäule}$«)

Nach der Konfiguration des nicht-terminalen Knotens werden die geänderten Bereiche erneut transformiert; um welche Bereiche es sich hierbei handelt, wird bei einem Abgleich zwischen Knoten und Prozessaktivitäten ermittelt. Da es für die Prozessaktivität »MRT-Bild bereitstellen« keinen Knoten im SPF-Graphen mehr gibt, wird sie aus der Instanz entfernt. V_{Trans} besteht nun lediglich aus dem neuen Knoten »Myelographie«.

Prozessaktivität »MRT-Bild bereitstellen« entfernen:

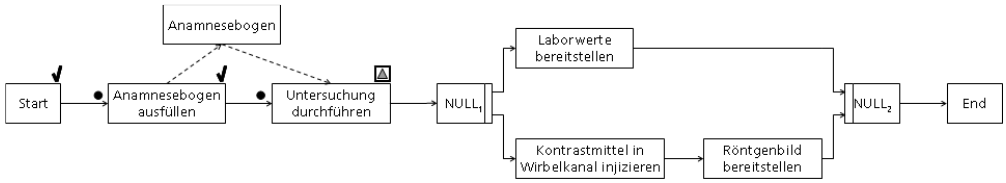
deleteActivity(W^I, »MRT-Bild bereitstellen«)

Knoten »Myelographie« transformieren:

N_{Before} = {Anamnesebogen ausfüllen, Untersuchung durchführen}
N_{After} = {Röntgenuntersuchung}
serialInsert(W^I, »$NULL_1$«, »Röntgenbild bereitstellen«, »Kontrastmittel in Wirbelkanal injizieren«, Implementation^{p6})

Abbildung 131 zeigt die Instanz nach der Rekonfiguration.

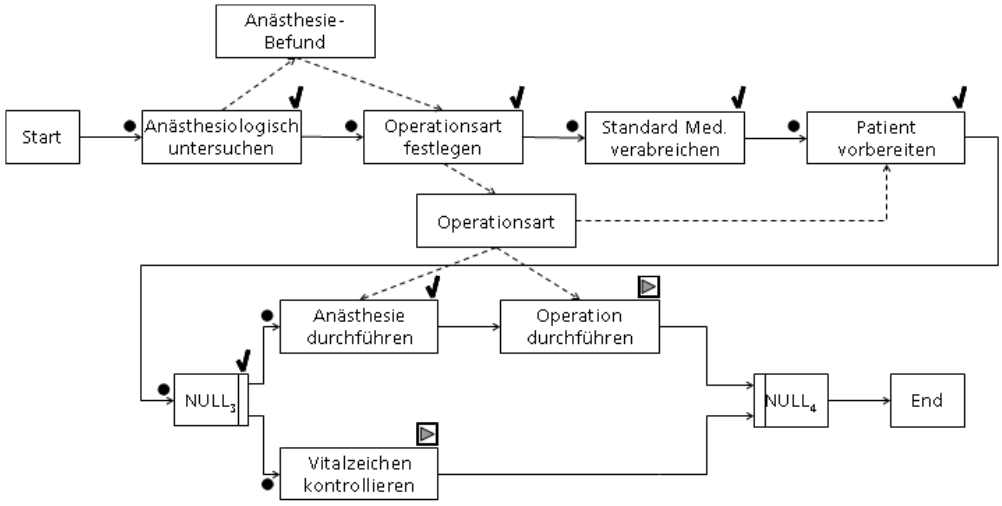
Abbildung 131 Instanz des Diagnostikprozesses nach der Rekonfiguration des SPF-Graphen und erneuter Transformation



Das sequentielle Einfügen des Prozessfragments »Kontrastmittel in Wirbelkanal injizieren« ist möglich, da sich die Bereiche N_{Before} und N_{After} nicht überschneiden.

Eine weitere Varianz bezieht sich auf das Auftreten einer Duraläsion während des Therapieprozesses, die intra- oder postoperativ behandelt werden kann (siehe Kapitel 2.2.1.4). Die folgende Abbildung zeigt, in welchem Status sich die Instanz des Therapieprozesses zum Zeitpunkt der Duraläsion befindet.

Abbildung 132 Instanz des klinischen Pfades zur Therapie von Wirbelsäulenerkrankungen



Das Auftreten dieser Komplikation wurde im SPF-Typ-Graphen bereits berücksichtigt. So existiert nun auch im SPF-Graphen in Abbildung 123 der Knoten »Komplikation«. Stellt der Operateur eine Duraläsion fest, so ist die Rekonfiguration dieses Knotens notwendig, um die Reaktion auf die Komplikation prozessorientiert zu dokumentieren. In diesem Fall beinhaltet die Rekonfiguration lediglich das Einfügen des Knotens »Versorgung Duraläsion« unterhalb des Rekonfigurationsknotens »Komplikation«:

$$addNodeRuntime(STG, SG^{I}, I, »Komplikation«, »N_{Versorgung\ Duraläsion}«)$$

Durch automatisierte Konfiguration wird der SPF-Knoten nach dem Einfügen direkt in sein terminales Symbol umgewandelt. Anschließend wird der Knoten transformiert mit $V_{Trans} = \{Versorgung\ Duraläsion\}$. Neben den Prozessfragmenten, deren Knoten im SPF-Graphen über Constraints mit dem einzufügenden Knoten oder seinen Vorgängern verbunden sind, umfasst die Menge N_{Before} nun auch alle diejenigen Aktivitäten, deren Ausführung begonnen hat oder bereits abgeschlossen ist.

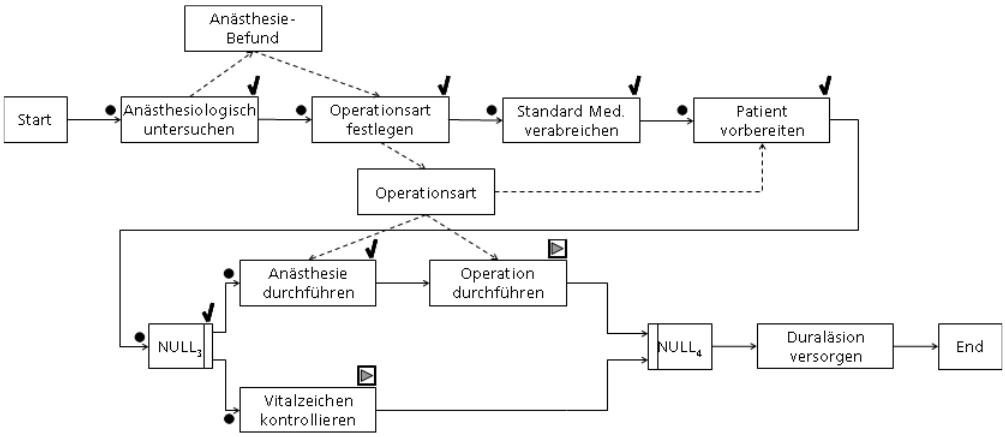
Knoten »Versorgung Duraläsion« transformieren:

N_{Before} = *{Anamnesebogen ausfüllen, Untersuchung durchführen, Standard Medikation verabreichen, Patient vorbereiten, Anästhesie durchführen, Vitalzeichen kontrollieren, Operation durchführen}*

N_{After} = ∅

serialInsert(W^I, »$NULL_4$«, »End«, »Duraläsion versorgen«, *Implementation*^{p8})

Abbildung 133 Instanz des Therapieprozesses nach der Rekonfiguration des SPF-Graphen und erneuter Transformation



Das neue Fragment wird seriell hinter dem parallelen Block in die Prozessinstanz eingefügt, da sich die Aktivitäten »Operation durchführen« und »Vitalzeichen kontrollieren« bereits im Zustand »Running« befinden.

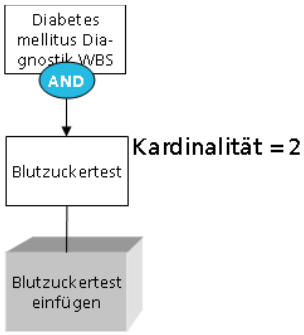
Die manuelle Rekonfiguration bietet sich auch für die Varianz der thromboembolischen Komplikation (siehe Kapitel 2.2.1.3) an. Die Behandlung dieser Varianz findet in einem eher inkrementellen Vorgehen statt; d. h. mit jedem Schritt wird der Erkenntnisstand über das Vorliegen und die konkrete Art der Komplikation aktualisiert. Ausgehend von den neuen Erkenntnissen wird dann über den nächsten zu treffenden Schritt entschieden. Da diese Varianz allerdings nicht während Diagnostik und Therapie, sondern erst im Laufe der stationären Nachsorge auftritt, soll ihre detaillierte Darstellung an dieser Stelle ausgeklammert werden.

5.5 Prozessvariante »Diabetes mellitus«

Bei der Untersuchung des Phänomens der Varianz von klinischen Pfaden in Kapitel 2.2 wurde festgestellt, dass neben eher einfachen Änderungen, wie z. B. dem Entfernen der MRT-Untersuchung und dem Einfügen einer Myelographie oder dem zusätzlichen Versorgen einer Duraläsion, auch komplexe Anpassungen erforderlich sein können. Die manuelle Rekonfiguration ist in solchen Fällen meist zu aufwändig und fehleranfällig. Darüber hinaus macht es oft Sinn, auch bei Abweichungen Standards vorzugeben, die dem Personal den Umgang mit der Varianz erleichtern sowie die vorausschauende Kostenkalkulation und die Berücksichtigung von evidenzbasierten Erkenntnissen ermöglichen. Aus diesem Grund sollen für solche Arten von Varianz Prozessvarianten definiert werden, auf Basis derer klinische Pfade im Sinne eines »best practice« Ansatzes weitgehend automatisch modifiziert werden können. In diesem Kapitel soll die Umsetzung der Prozessvariante »Diabetes mellitus« (siehe Kapitel 2.2.1.2) vorgestellt werden. Mit Hilfe der Prozessvariante »Diabetes mellitus« lassen sich die klinischen Pfa-

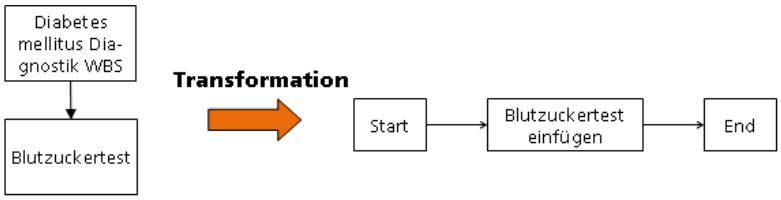
de für Diagnostik und Therapie von Wirbelsäulenerkrankungen so verändern, dass sie für die Behandlung von Diabetikern geeignet sind. Da diese Variante z. B. im Hinblick auf die Anzahl an Blutzuckertests und die Notwendigkeit einer Antibiose variabel gestaltet werden kann, wird jeweils ein SPF-Typ-Graph für sie erstellt, wobei sich der eine auf den SPF-Typ-Graphen für die Diagnostik und der andere auf den SPF-Typ-Graphen für die Therapie von Wirbelsäulenerkrankungen bezieht. Die nächste Abbildung zeigt den SPF-Typ-Graph der Prozessvariante Diabetes mellitus zur Manipulation des Diagnostikprozesses.

Abbildung 134 SPF-Typ-Graph der Prozessvariante »Diabetes mellitus« für die Diagnostik von Wirbelsäulenerkrankungen



Gemäß diesem SPF-Typ-Graphen bewirkt die Prozessvariante zwingend das Einfügen eines Blutzuckertests. Die Variante ist jedoch konfigurierbar, da wahlweise auch ein zweiter Test vorgenommen werden kann. Der Standard der Prozessvariante sieht jedoch nur einen Blutzuckertest vor. Die nächste Abbildung zeigt den Standard SPF-Graph für die Variante so wie die Prozessdefinition nach der Transformation.

Abbildung 135 SPF-Graph und Prozessdefinition der Prozessvariante »Diabetes mellitus« für die Diagnostik von Wirbelsäulenerkrankungen



Die Implementierung des Prozessfragments basiert auf dem Aufruf der SPF-Änderungsoperation INSERT; diese wird nun zur Modifikation des SPF-Graphen in Abbildung 122 eingesetzt:

spfInsert(STG, SG^I, W^I, Conditions, »$N_{Blutzuckertest}$«, ∅) mit *Conditions = {ALL, Laboruntersuchung,* ∅ , ∅ , ∅ *)*

Da es sich bei Blutzuckertest um einen SPF-Knoten handelt, muss keine temporäre Graphgrammatik angegeben werden, um den Knoten zu konfigurieren; dies geschieht ohnehin durch automatisierte Konfiguration (siehe Kapitel 4.4.4.1). Die erneute Transformation mit V_{Trans} = *{Blutzuckertest}* wird nun ausgehend von der Prozessinstanz in Abbildung 131 durchgeführt.

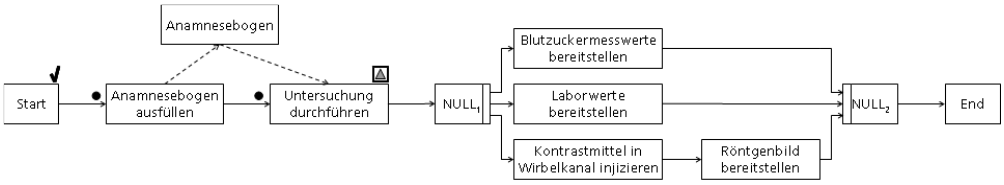
Knoten »Blutzuckertest« einfügen:

N_{Before} = *{Anamnesebogen ausfüllen, Untersuchung durchführen}*

N_{After} = ∅

parallelInsert(W^{I}, »$NULL_1$«, »$NULL_2$«, »Blutzuckermesswerte bereitstellen«, Implementation^{P7})

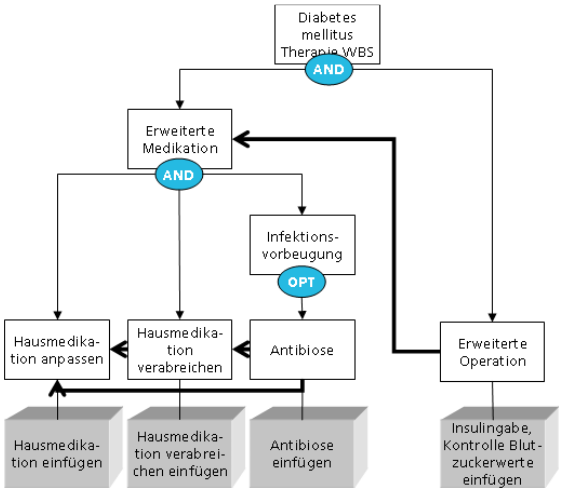
Abbildung 136 Prozessinstanz zur Diagnostik von Wirbelsäulenerkrankungen nach Aufruf der Prozessvariante »Diabetes mellitus«



Die Prozessvariante kann höchstens zweimal auf den klinischen Pfad bzw. seine Instanzen angewendet werden, da der SPF-Typ-Graph des Pfades nur maximal zwei Blutzuckertests erlaubt. Das wiederholte Aufrufen der Prozessvariante führt jedoch nicht zu Fehlern, sondern bleibt wirkungslos. An diesem Beispiel wird deutlich, dass sich die manuelle Rekonfiguration und die Auswahl von Prozessvarianten problemlos miteinander kombinieren lassen. Dennoch kann es vorkommen, dass bestimmte SPF-Änderungsoperationen einer Variante in Folge einer Rekonfiguration nicht mehr anwendbar sind, weil z. B. der Rekonfigurationsknoten, an dem die Operation ansetzt, gelöscht wurde. Dies widerspricht jedoch nicht der Semantik einer vordefinierten ad hoc Änderung, da diese nur innerhalb eines bestimmten Anwendungskontextes stattfinden kann; ist dieser Kontext nicht mehr gegeben, macht auch die Modifikation keinen Sinn.

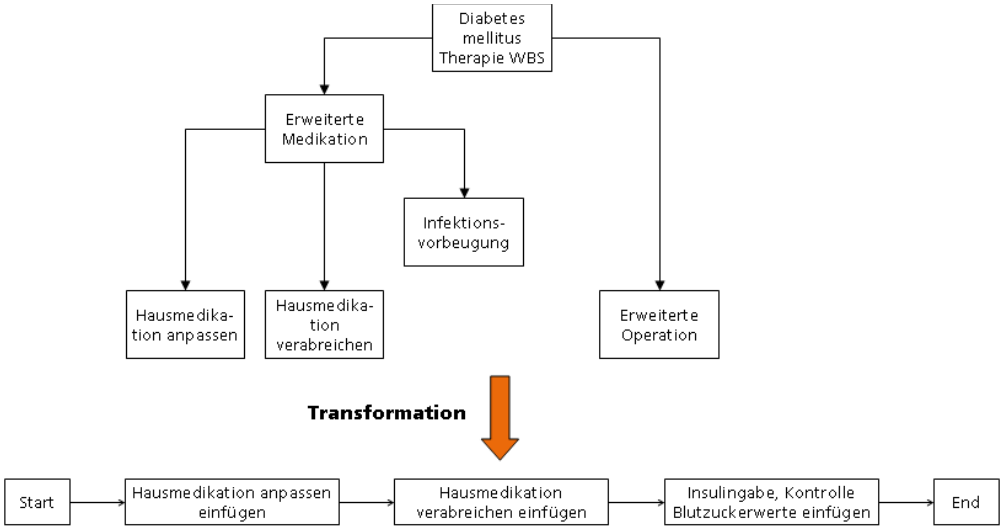
Etwas komplexer ist die Prozessvariante »Diabetes mellitus« zur Anpassung von Therapieprozessen. Die folgende Abbildung zeigt den SPF-Typ-Graph der Variante.

Abbildung 137 SPF-Typ-Graph der Prozessvariante »Diabetes mellitus« für die Therapie von Wirbelsäulenerkrankungen



Dieser SPF-Typ-Graph hat zur Folge, dass die Knoten »Hausmedikation«, »Hausmedikation verabreichen«, »Insulingabe« und »Kontrolle Blutzuckerwerte« in den SPF-Graphen des klinischen Pfades eingefügt werden. Da die Knoten »Insulingabe« und »Kontrolle Blutzuckerwerte« unterhalb desselben Rekonfigurationsknotens hinzugefügt werden, ist für den Einfügevorgang nur ein Prozessfragment erforderlich. Optional kann die Prozessvariante auch den Knoten »Antibiose« umfassen. Da dieser jedoch nur dann in den Prozess eingefügt werden soll, wenn für den Diabetes Patienten wirklich ein erhöhtes Infektionsrisiko besteht, enthält die Standard Konfiguration dieses SPF-Typ-Graphen den Knoten nicht. Die folgende Abbildung zeigt den SPF-Graphen als Standard der Prozessvariante sowie die daraus abgeleitete Prozessdefinition.

Abbildung 138 SPF-Graph und Prozessdefinition der Prozessvariante »Diabetes mellitus« für die Therapie von Wirbelsäulenerkrankungen



Die Implementierung der Prozessfragmente basiert auf Aufrufen der SPF-Änderungsoperation INSERT; diese wird nun zur Modifikation des SPF-Graphen in Abbildung 123 eingesetzt:

spfInsert(STG, SG, W, Conditions, »$N_{Hausmedikation\ anpassen}$«, ∅) mit Conditions = {ALL, »Erweiterte Planung«, ∅ , ∅ , ∅ , ∅)

spfInsert(STG, SG, W, Conditions, »$N_{Erweiterte\ Medikation}$«, temp-GG) mit Conditions = {ALL, »Medikation«, ∅ , ∅ , ∅ , ∅) und temp-GG = {$N_{Erweiterte\ Medikation} \rightarrow N^1_{Erweiterte\ Medikation}$, $N^1_{Erweiterte\ Medikation} \rightarrow$ Erweiterte Medikation($N_{Hausmedikation\ verabreichen}$), $N_{Hausmedikation\ verabreichen} \rightarrow N^1_{Hausmedikation\ verabreichen}$, $N^1_{Hausmedikation\ verabreichen} \rightarrow$ Hausmedikation verabreichen}

spfInsert(STG, SG, W, Conditions, (»$N_{Insulingabe}$«, »$N_{Kontrolle\ Blutzuckerwerte}$«), ∅) mit Conditions = {ALL, »Erweiterte Maßnahmen«, ∅ , ∅ , ∅ , ∅)

Die Transformation des SPF-Graphens nach Anwendung der Variante mit V_{Trans} = *{Hausmedikation anpassen, Hausmedikation verabreichen, Insulingabe, Kontrolle Blutzuckerwerte}* wird nun ausgehend von dem ADEPT2 WSM Net in Abbildung 129 durchgeführt.

Knoten »Hausmedikation anpassen« transformieren:

N_{Before} = {Anästhesiologisch untersuchen}
N_{After} = {Standard Medikation verabreichen, Patient vorbereiten, Anästhesie durchführen, Operation durchführen, Vitalzeichen kontrollieren}
parallelInsert(W, »Operationsart festlegen«, »Operationsart festlegen«, »Hausmedikation anpassen« Implementation^{p8})

Knoten »Hausmedikation verabreichen« transformieren:

N_{Before} = {Anästhesiologisch untersuchen, Hausmedikation anpassen, Operationsart festlegen}
N_{After} = {Patient vorbereiten, Anästhesie durchführen, Operation durchführen, Vitalzeichen kontrollieren}
parallelInsert(W, »Standard Medikation verabreichen«, »Standard Medikation verabreichen«, »Hausmedikation verabreichen« Implementation^{p9})

Knoten »Insulingabe« transformieren:

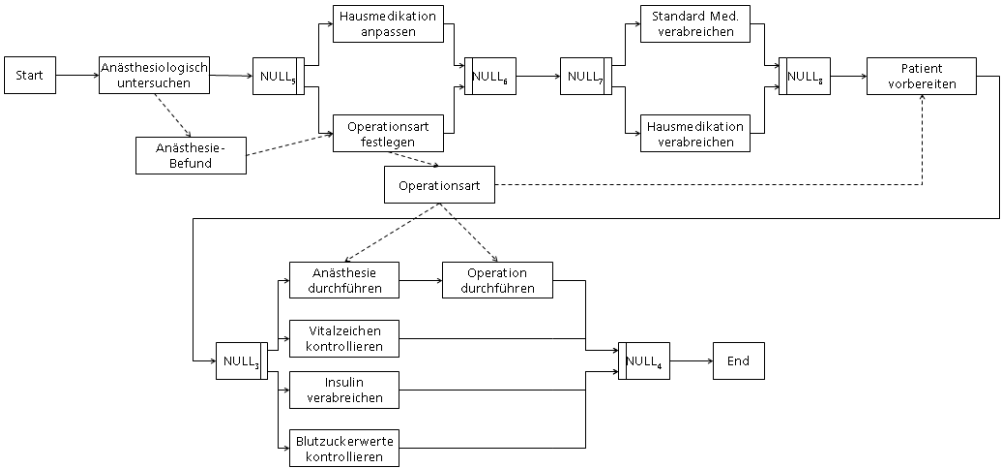
N_{Before} = {Anästhesiologisch untersuchen, Hausmedikation anpassen, Operationsart festlegen, Standard Medikation verabreichen, Hausmedikation verabreichen, Patient vorbereiten}
N_{After} = ∅
parallelInsert(W, »$NULL_3$«, »$NULL_4$«, »Insulin verabreichen« Implementation^{p10})

Knoten »Kontrolle Blutzuckerwerte« transformieren:

N_{Before} = {Anästhesiologisch untersuchen, Hausmedikation anpassen, Operationsart festlegen, Standard Medikation verabreichen, Hausmedikation verabreichen, Patient vorbereiten}
N_{After} = ∅
parallelInsert(W, »$NULL_3$«, »$NULL_4$«, »Blutzuckerwerte kontrollieren« Implementation^{p11})

Die folgende Abbildung illustriert die resultierende Prozessdefinition.

Abbildung 139 Prozessdefinition zur Therapie von Wirbelsäulenerkrankungen nach Aufruf der Prozessvariante »Diabetes mellitus«



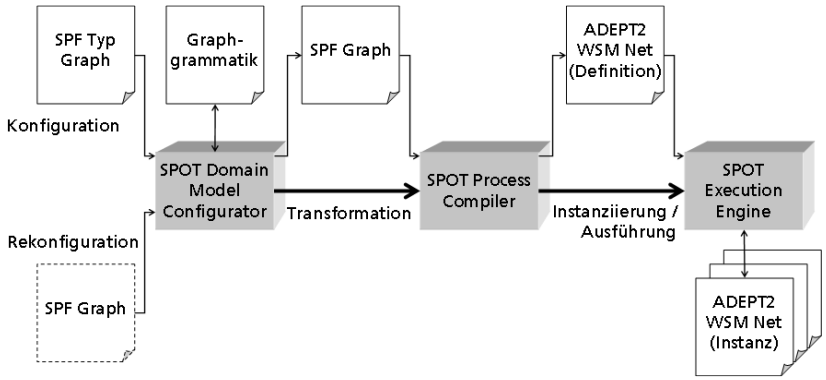
6 Prototypische Umsetzung

Während das Lösungskonzept an Hand einer Fallstudie aus dem medizinischen Bereich in Kapitel 5 theoretisch validiert wurde, soll in diesem Abschnitt seine praktische Umsetzung vorgestellt werden. Damit lässt sich nachweisen, dass das Lösungskonzept auch IT-technisch implementiert werden kann. Die Realisierung des Prototypens wird in [Rudolph 2010] ausführlich beschrieben; daher sollen hier lediglich die wesentlichen Grundzüge vorgestellt werden. Die prototypische Implementierung lässt sich optimal in die Architekturlandschaft von SPOT eingliedern. Sie umfasst drei Komponenten:

- Der **SPOT-Domain-Model-Configurator** dient dem Erzeugen von Graphgrammatiken auf Basis von SPF-Typ-Graphen sowie der Ableitung von SPF-Graphen durch Konfiguration. Er setzt alle in Kapitel 4.4 beschriebenen Konzepte praktisch um. Darüber hinaus ermöglicht er die manuelle Rekonfiguration wie in Kapitel 4.6 dargestellt, wobei er sowohl den strukturerhaltenden als auch den nicht-strukturerhaltenden Ansatz unterstützt. Der SPOT Domain Model Configurator entspricht in seiner aktuellen Form einer Alternative zum SPOT-Care-Plan-Modeler (siehe Kapitel 4.1.1); da Werkzeuge, die eine für Anwender optimierte DSL unterstützen, bei einer Erprobung des Lösungskonzeptes in einer praktischen Umgebung jedoch unverzichtbar sind, muss hier mittelfristig eine geeignete Synthese erfolgen.
- Der bereits in Kapitel 4.1 skizzierte **SPOT-Process-Compiler** wurde so erweitert, dass er ausgehend von einem SPF-Graphen und dem entsprechenden SPF-Typ-Graphen ADEPT2 WSM Nets erzeugt. Damit realisiert er die im Rahmen der Transformationsphase entwickelten Spezifikationen (siehe Kapitel 4.5). Darüber hinaus ermöglicht er die erneute Transformation von Prozessdefinitionen nach einer Rekonfiguration zur Modellierungszeit.
- Als **SPOT Execution Engine** wird die Aristaflow® BPM Suite genutzt. Mit ihr lassen sich die vom SPOT-Process-Compiler erzeugten und gegebenenfalls bereits rekonfigurierten Prozessdefinitionen sofort ausführen. Damit die Aktivitäten der Prozessdefinition bearbeitet werden können, wurden für alle benötigten Prozessfragmente Aktivitätsvorlagen erzeugt und in das Repository des WfMS eingestellt.

Die nächste Abbildung verdeutlicht die Gesamtarchitektur des Prototypens bestehend aus diesen drei Komponenten.

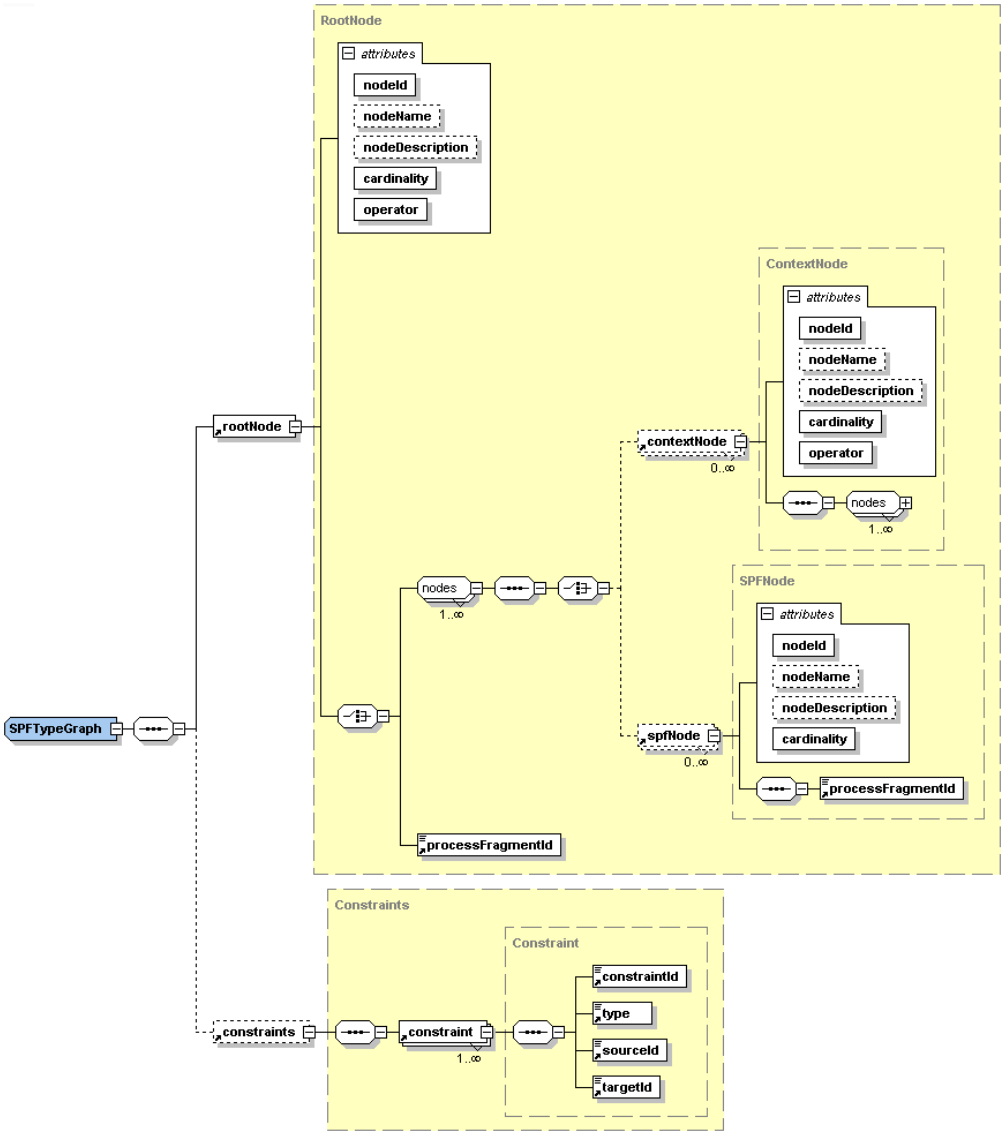
Abbildung 140 Übersicht über die Architektur des Prototypen



Im Kontext der Implementierung wurde kein Werkzeug entwickelt, das die Erstellung von SPF-Typ-Graphen als Domänenmodelle für klinische Pfade und Prozessvarianten ermöglicht. Es wurde jedoch ein XML-Schema für SPF-Typ-Graphen erstellt, das auch bereits einige der in Kapitel 4.3.6 spezifizierten Korrektheitskriterien abdecken kann. Alle SPF-Typ-Graphen entsprechen daher XML Dokumenten, die noch manuell kreiert werden müssen. Die erzeugten XML-Dokumente können dann vom SPOT-Domain-Model-Configurator eingelesen und dargestellt werden. Die Umsetzung von Prozessvarianten ist ebenfalls nicht Bestandteil des aktuellen Prototypen; die dafür notwendigen Erweiterungen beziehen sich auf die Identifikation von Knoten im SPF-Graphen (siehe Kapitel 4.6.3), die Implementierung der SPF-Änderungsoperationen (siehe Kapitel 4.7.1), sowie die Umsetzung des Top-Level Prozesses zur Auswahl und Anwendung von Prozessvarianten (siehe Kapitel 4.7.2). Für die Konfiguration von Prozessvarianten basierend auf SPF-Typ-Graphen und die Transformation in ausführbare Prozessdefinitionen können hingegen die bereits entwickelten Komponenten SPOT-Domain-Model-Configurator und SPOT-Process-Compiler genutzt werden.

Um einen SPF-Graphen zu erzeugen, muss zunächst ein SPF-Typ-Graph als XML-Dokument vorliegen, das konform zum XML-Schema für SPF-Typ-Graphen ist (siehe Abbildung 141); in Anhang A.1 kann das Schema in textueller Form eingesehen werden.

Abbildung 141 XML Schema für SPF-Typ-Graphen



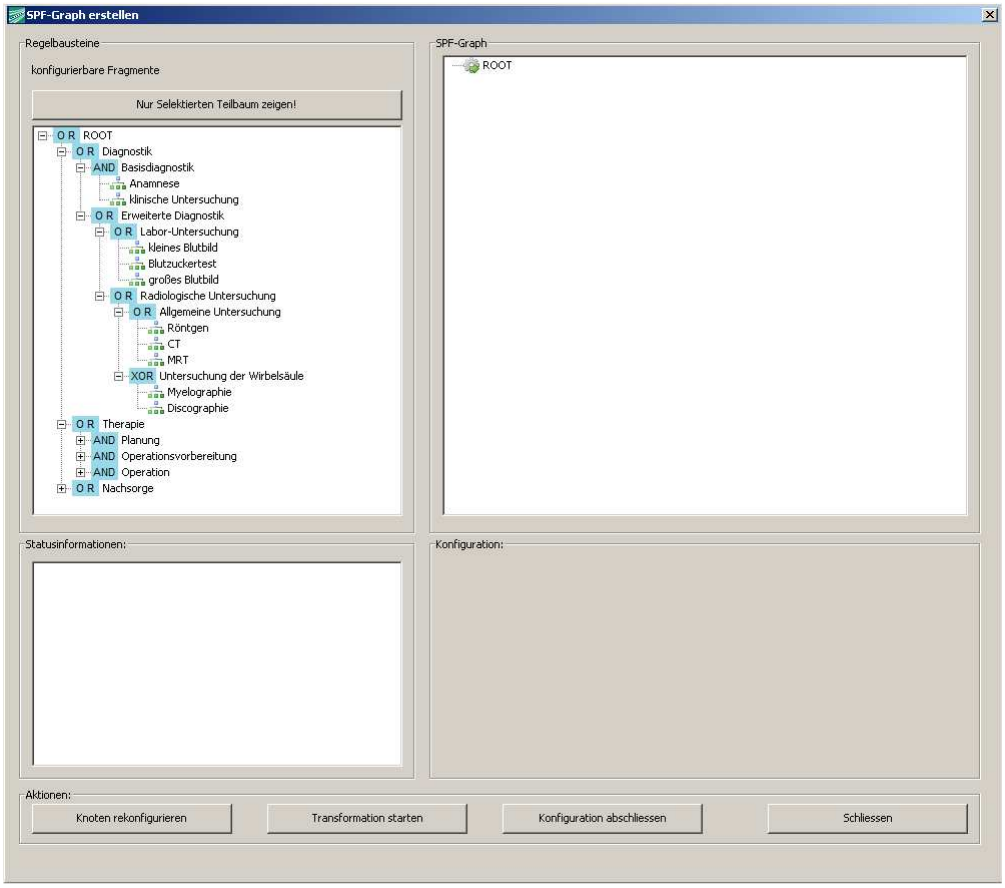
Gemäß Schema besteht jeder SPF-Typ-Graph aus zwei Bereichen: Einer Menge von Knoten und einer Menge von Constraint-Relationen. Alle Knoten erben ihre wesentlichen Eigenschaften vom abstrakten Typ »Node«; dieser definiert z. B., dass jeder Knoten über eine eindeutige Identität verfügen muss. Der abstrakte Typ »Node« wird verfeinert durch die konkreten Datentypen »RootNode«, »ContextNode« und »SPFNode«. Auf diese Weise lassen sich für die verschiedenen Arten von Knoten zusätzliche Merkmale festlegen, die den jeweiligen Knotentyp auszeichnen.

Ein valider SPF-Typ-Graph umfasst mindestens den Wurzelknoten, der selbst keine übergeordneten Knoten hat. Der Wert der Kardinalität ist mit dem Wert eins für den Wurzelknoten fest voreingestellt. Innerhalb des Elements »rootNode« kann es beliebig viele Elemente »contextNode« bzw. »spfNode« geben; im Gegensatz dazu kann die Wurzel auch lediglich einen Verweis auf ein Prozessfragment umfassen. Beide Optionen schließen sich gegenseitig aus. Die Struktur des XML-Schemas gewährleistet damit die Einhaltung sämtlicher Wurzelbe-

dingungen aus Kapitel 4.3.6. Alle SPF-Knoten besitzen zwingend die Zuordnung zu einem Prozessfragment. Das XML-Schema dient so der Erfüllung der SPF-Knotenbedingung und damit auch der Bildung von semantischen Prozessfragmenten. Durch die im XML Schema vorgegebene strenge Schachtelungsstruktur wird automatisch sichergestellt, dass tatsächlich jeder Knoten von der Wurzel aus erreicht werden kann. Damit ist auch das Kriterium der Knotenerreichbarkeit erfüllt. Auch Zyklen kann es auf aufgrund dieser klassischen XML-Struktur nicht geben. Darüber hinaus lässt sich im XML-Schema einfach definieren, welche Typen von logischen Operatoren oder Constraint-Relationen angegeben werden dürfen und dass der Wertebereich der Kardinalitäten die positiven ganzen Zahlen sind. Die Einhaltung von Constraintbedingungen und Operator-Constraint-Kombinationsbedingungen hingegen lässt sich allein auf Basis des XML-Schemas nur sehr eingeschränkt sicherstellen. In dem XML-Schema für SPF-Typ-Graphen können aktuell unabhängig von der konkreten Menge an Knoten beliebig viele Constraints definiert werden. Die Elemente »sourceId« und »targetId« sollen sich auf die Attribute »nodeId« der definierten Knoten beziehen; dies wird vom Schema aktuell jedoch nicht automatisch kontrolliert. Darüber hinaus wird nicht garantiert, dass keine Constraints zwischen Knoten festgelegt werden, die über hierarchische Kanten miteinander verbunden sind usw. Die Überprüfung dieser Kriterien muss also über erweiterte Methoden, z. B. durch Nutzung von Xpath-Ausdrücken stattfinden [WWWC 2005]; dies ist derzeit jedoch nicht Gegenstand der prototypischen Implementierung.

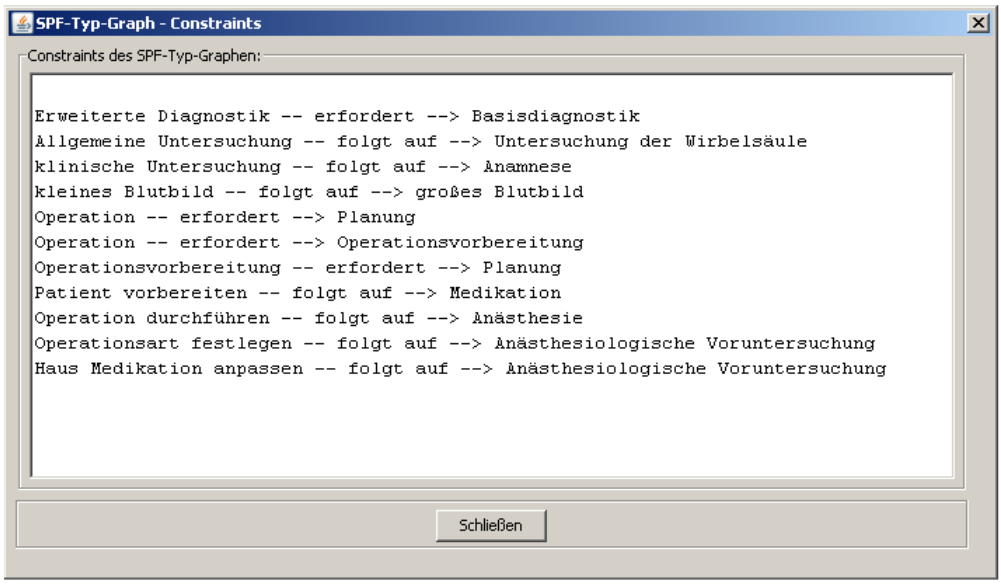
Der SPF-Typ-Graph für die Behandlung von Wirbelsäulenerkrankungen wurde als XML-Dokument auf Basis dieses Schemas erstellt; Anhang A.2 zeigt den Ausschnitt für die Diagnostik. Schema-konforme XML-Dokumente können vom SPOT-Domain-Model-Configurator eingelesen und dargestellt werden. Die nächste Abbildung zeigt den SPF-Typ-Graphen für die Behandlung von Wirbelsäulenerkrankungen im SPOT-Domain-Model-Configurator.

Abbildung 142 SPF-Typ-Graph für die Behandlung von Wirbelsäulenerkrankungen im SPOT-Domain-Model-Configurator



Während die logischen Operatoren direkt neben den Knotenbezeichnungen erscheinen, sind die Constraint-Relationen nicht in die Darstellung integriert. Diese lassen sich aktuell über ein separates Fenster abfragen.

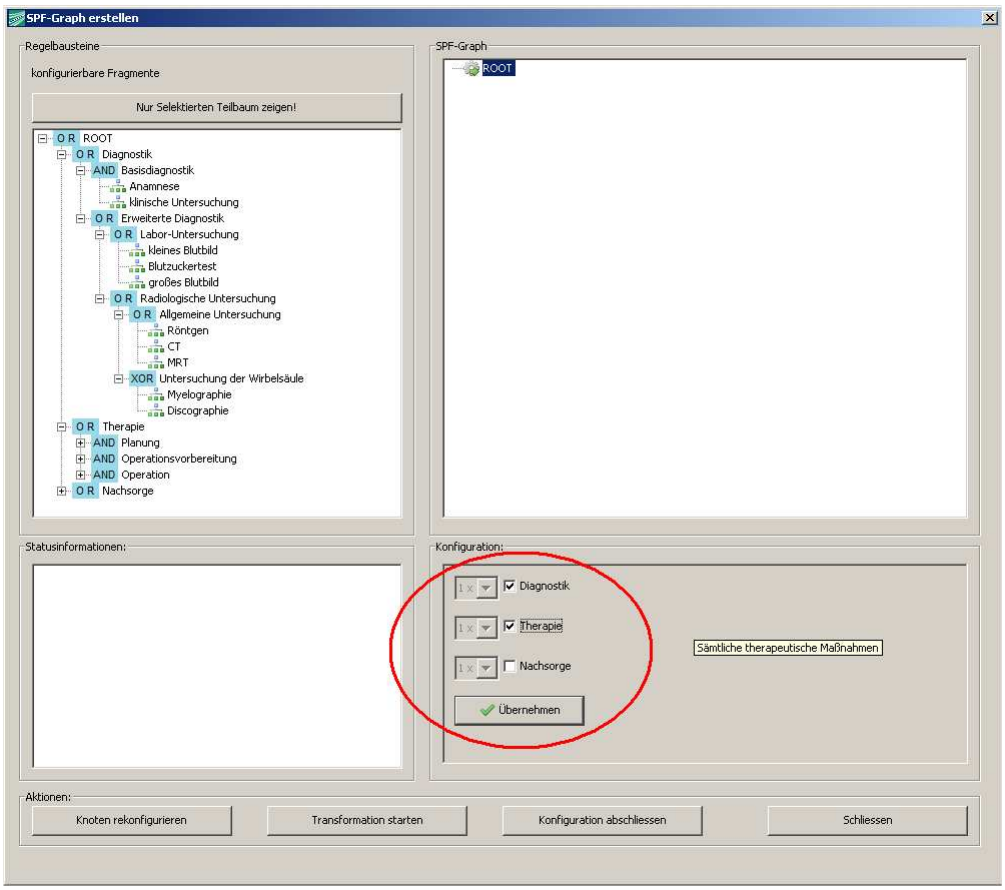
Abbildung 143 Übersicht über die im SPF-Typ-Graphen definierten Constraint-Relationen



Mit dem Einlesen und der Darstellung des SPF-Typ-Graphen ist auch die Generierung der Graphgrammatik verbunden, die die Produktionsregeln bereitstellt, durch deren Anwendung typisierte SPF-Graphen erzeugt werden können.

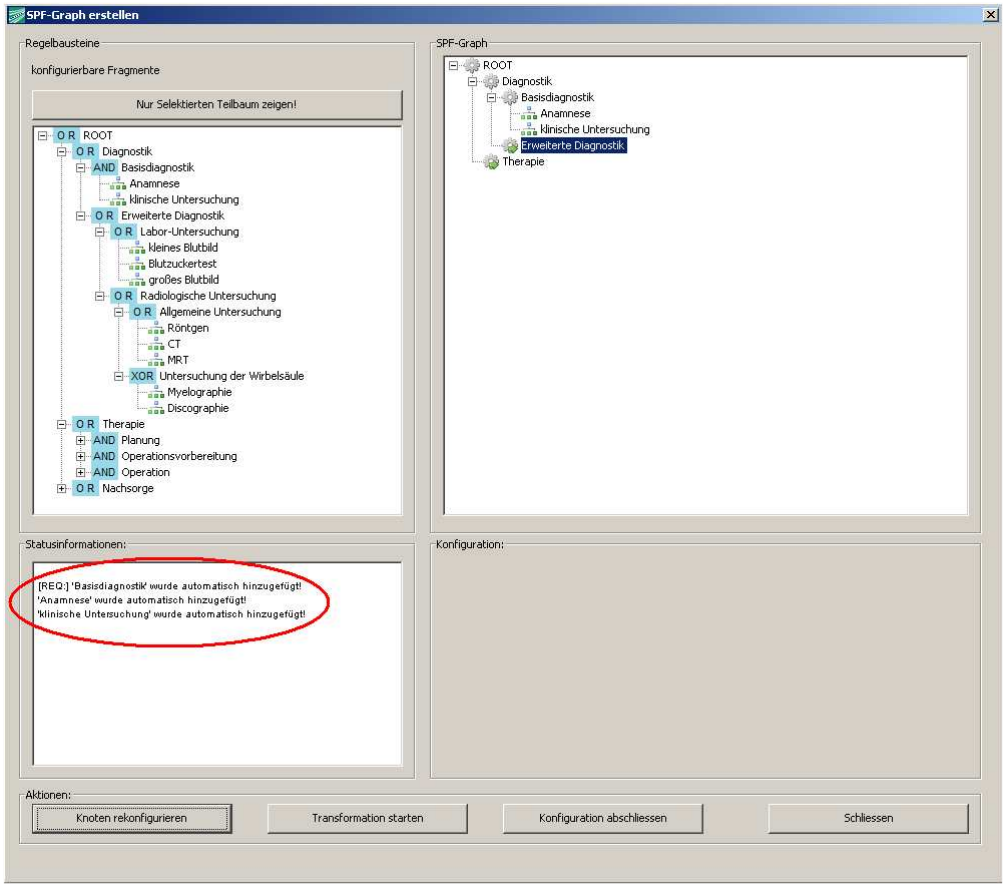
Während auf der linken Seite der Konfigurationsansicht in Abbildung 142 lediglich der SPF-Typ-Graph dargestellt wird, erscheint auf der rechten Seite der aktuell davon abgeleitete SPF-Graph. Dieser besteht initial aus dem Wurzelknoten. Da der Wurzelknoten im SPF-Typ-Graphen mit einem OR-Operator assoziiert ist, muss die erste Konfigurationsentscheidung manuell von der Fachanwenderin getroffen werden. Dazu wird der Wurzelknoten im SPF-Graphen selektiert. Anschließend erscheinen im unteren rechten Feld die Konfigurationsmöglichkeiten (siehe Abbildung 144). Fachanwender haben die Möglichkeit, festzulegen, ob der klinische Pfad Diagnostik, Therapie und/oder Nachsorge umfassen soll.

Abbildung 144 Konfiguration des Wurzelknotens im SPOT-Domain-Model-Configurator



Die Fachanwenderin entscheidet sich für die Einbettung von diagnostischen und therapeutischen Maßnahmen in den SPF-Graphen. Durch die Selektion der entsprechenden Felder und einen Klick auf die Schaltfläche »Übernehmen« wird die entsprechende Regel der Graphgrammatik angewendet, die dazu führt, dass unterhalb des Wurzelknotens nun die Knoten »Diagnostik« und »Therapie« erscheinen. Wie in der nachfolgenden Abbildung dargestellt, soll nun als nächstes der Knoten »Diagnostik« konfiguriert werden. Die Fachexpertin selektiert den Knoten »Erweiterte Diagnostik«. Da zwischen den Knoten »Erweiterte Diagnostik« und »Basisdiagnostik« im SPF-Typ-Graphen jedoch ein Anforderungs-Constraint definiert ist (siehe Abbildung 143), bewirkt diese Auswahl der Fachanwenderin nicht nur den Einschluss der selektierten Knoten, sondern auch der durch den Constraint erzwungenen Knoten, wie die nächste Abbildung zeigt.

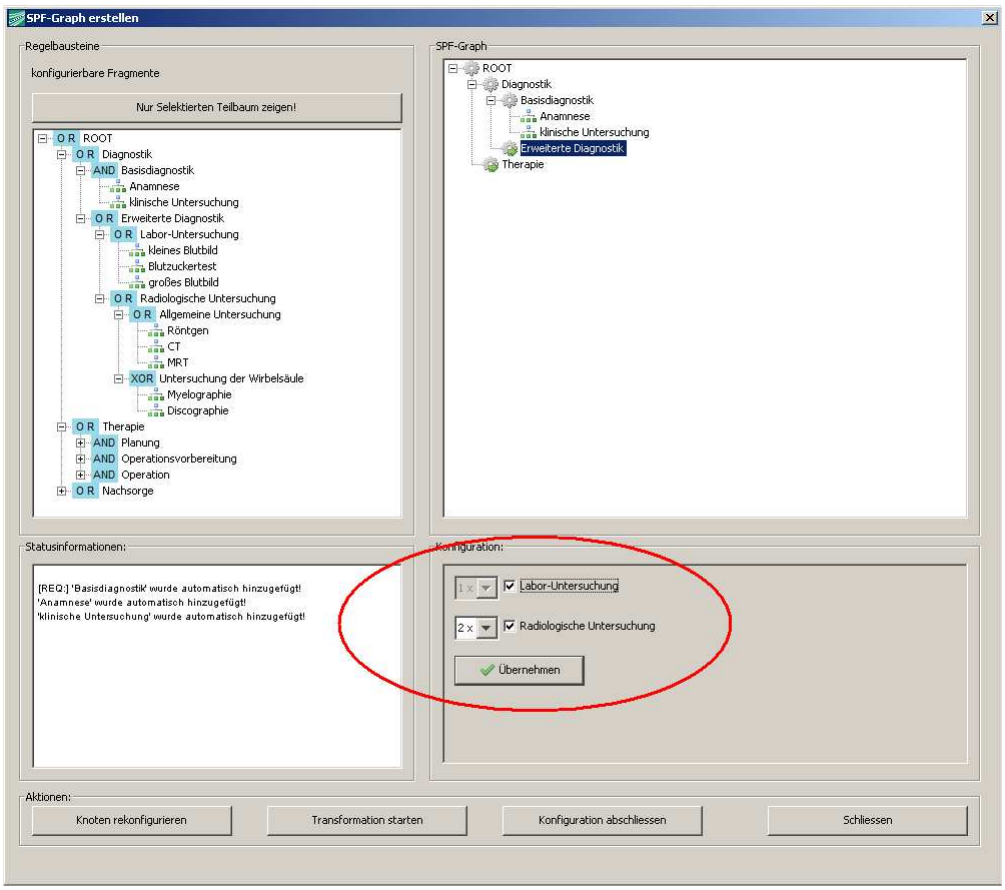
Abbildung 145 Durchsetzung von Constraints und automatisierte Konfiguration von Knoten im SPOT-Domain-Model-Configurator



Die Anwendung nutzt die in Kapitel 4.4.3.2 vorgestellten Algorithmen zur Identifikation einer Constraint-Verletzung sowie zur strukturerhaltenden Rekonfiguration des Knotens »Diagnostik«. Durch die Rekonfiguration wird sichergestellt, dass der Constraint-relevante Knoten »Basisdiagnostik« nun auch in der Konfiguration enthalten ist. Darüber hinaus ist aus der Abbildung ersichtlich, dass nicht nur der Knoten »Basisdiagnostik« eingefügt wurde, sondern auch dessen Nachfolger »Anamnese« und »klinische Untersuchung«. Ursache dafür ist die automatisierte Konfiguration aufgrund der Tatsache, dass dem Knoten »Basisdiagnostik« ein AND-Operator zugeordnet ist und es deswegen keine Konfigurationsalternativen gibt. In dem linken, unteren Feld »Statusinformationen« wird die Fachanwenderin über diese Zusammenhänge informiert.

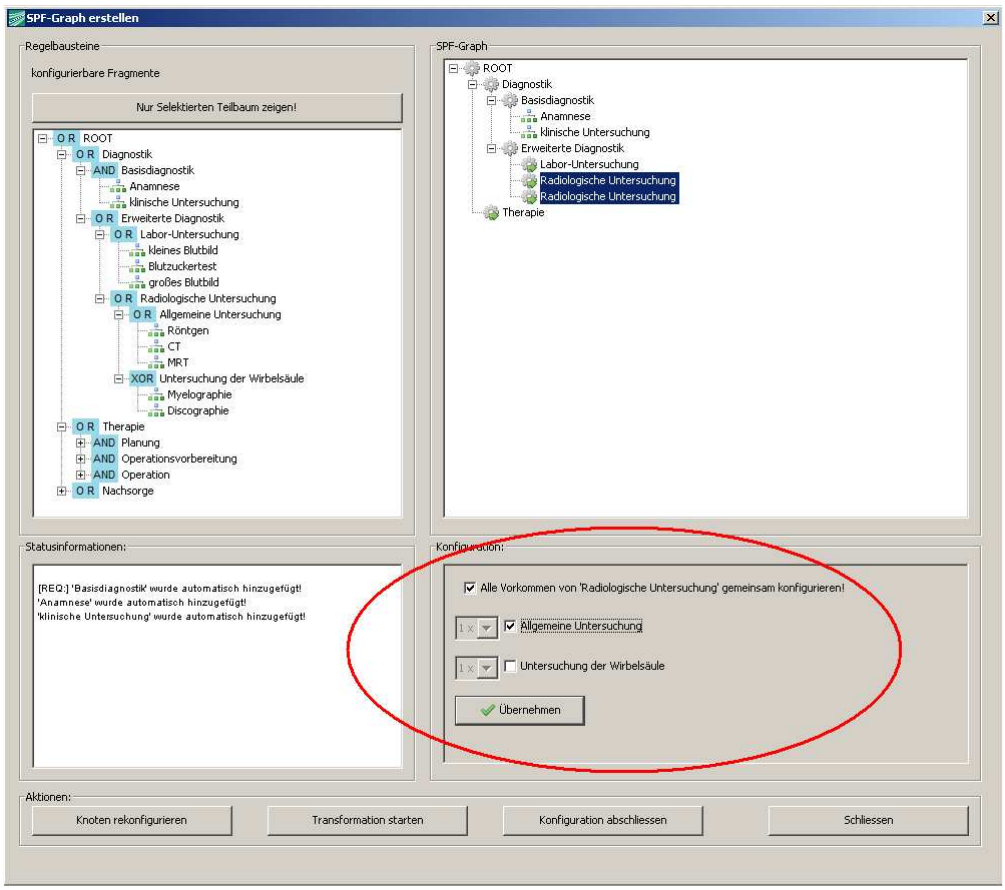
Im Rahmen der erweiterten Diagnostik sollen zwei radiologische Untersuchungen stattfinden. Dementsprechend entscheidet sich die Fachexpertin für das Anlegen zweier Klone vom Typ »Radiologische Untersuchung«.

Abbildung 146 Auswahl von Klonknoten im SPOT-Domain-Model-Configurator



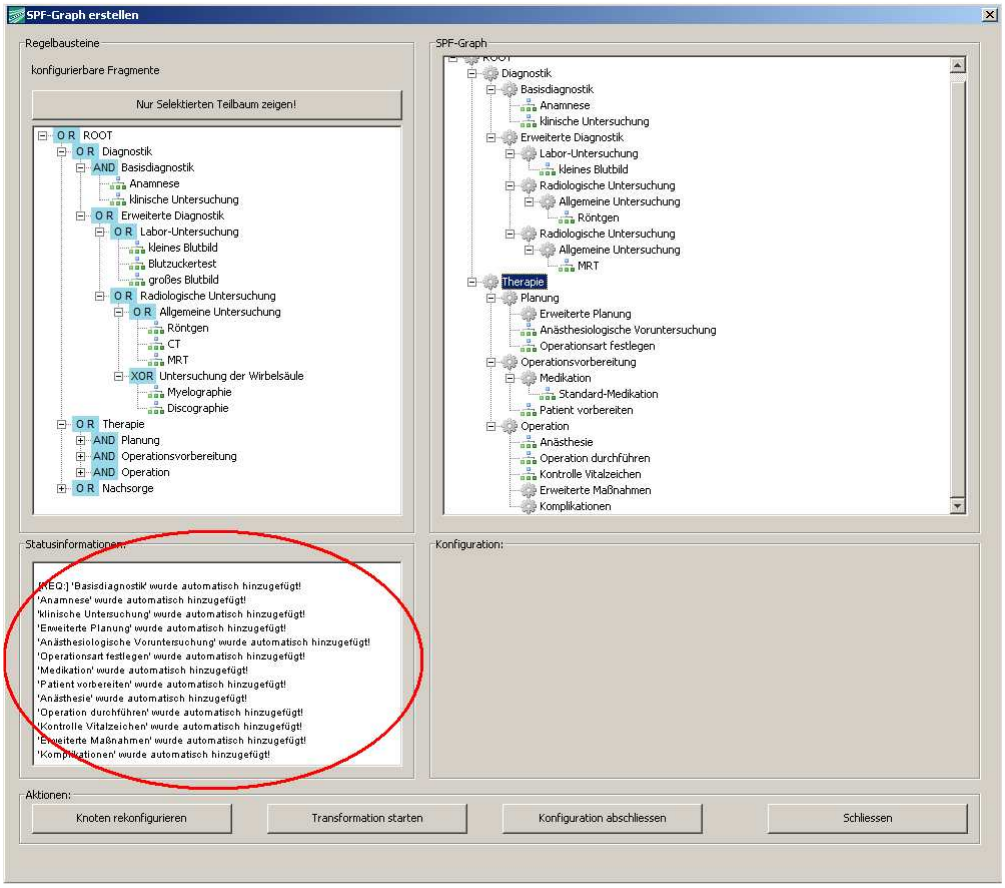
Anschließend sollen die beiden radiologischen Knoten so konfiguriert werden, dass sie lediglich aus allgemeinen Untersuchungen bestehen. Da dies für beide Klone gelten soll, können die Knoten gemeinsam konfiguriert werden, wie in der nächsten Abbildung dargestellt ist.

Abbildung 147 Gemeinsame Konfiguration von Klonknoten im SPOT-Domain-Model-Configurator



Die prototypische Implementierung setzt bisher nicht den erweiterten Algorithmus zur Identifikation von Knoten um; es besteht jedoch die Möglichkeit, gemäß Abschnitt 4.4.4.2 alle Klone eines Knoten automatisch auszuwählen und anschließend gleichartig zu konfigurieren. Die nächste Abbildung illustriert den vollständig konfigurierten SPF-Typ-Graphen, der nun den klinischen Pfad für die stationäre Behandlung von Wirbelsäulenerkrankungen repräsentiert.

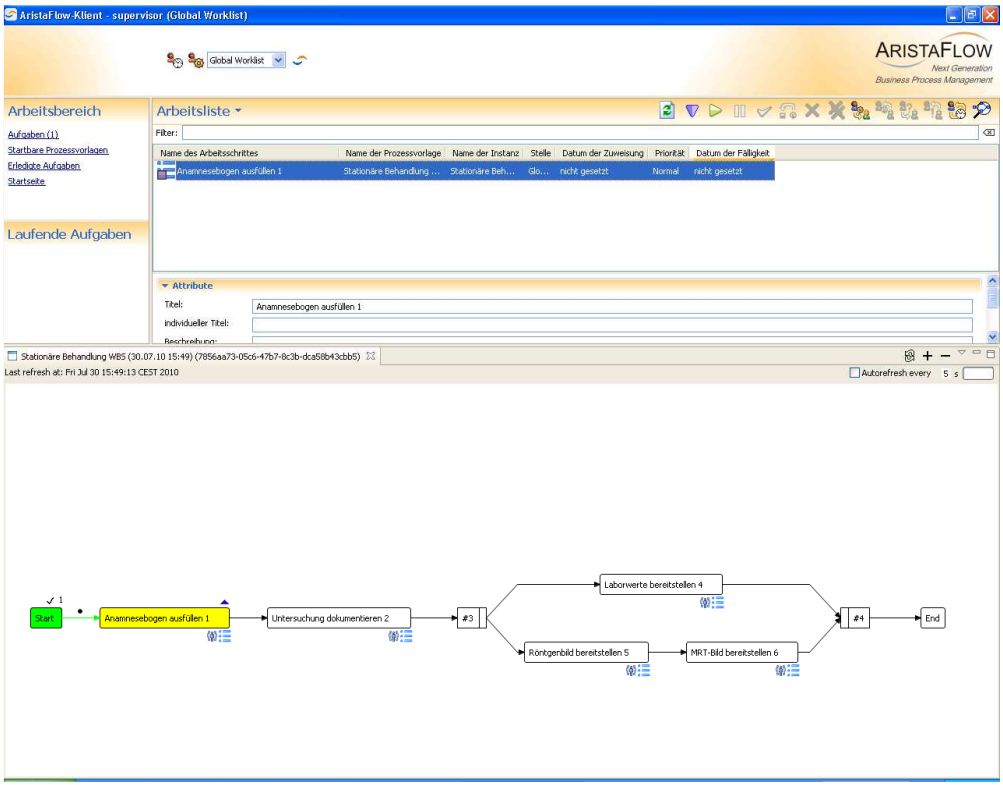
Abbildung 148 Vollständige Konfiguration des SPF-Typ-Graphen für die Behandlung von Wirbelsäulenerkrankungen im SPOT-Domain-Model-Configurator



Wie man an Hand der Statusinformationen erkennen kann, konnte der komplette Teilbaum ausgehend von der Wurzel »Therapie« fast vollständig automatisch generiert werden. Der Konfigurationsaufwand war also sehr gering. Der SPF-Typ-Graph ist nun vollständig konfiguriert, da allen noch nicht konfigurierten Kontextknoten OPT-Operatoren zugeordnet sind; diese werden vor der Transformation automatisch auf das jeweilige terminale Symbol abgebildet. Die manuelle Rekonfiguration einzelner Knoten und Teilgraphen unterscheidet sich in ihrer Darstellung und Durchführung für die Fachanwender nicht von der normalen Konfiguration.

Die so erzeugte XML-Repräsentation des SPF-Graphen (siehe Anhang A.2 für den Abschnitt Diagnostik) wird in einem nächsten Schritt dem SPOT-Process-Compiler übergeben, der gemäß den Transformationsregeln aus Kapitel 4.5.3 einen ausführbare Prozessdefinition erzeugt. Die letzte Abbildung zeigt eine Prozessinstanz des erzeugten klinischen Pfades für die Phase Diagnostik im Kontext der stationären Behandlung von Wirbelsäulenerkrankungen im Aristaflow® Client.

Abbildung 149 Prozessinstanz des klinischen Pfades für die Diagnostik von Wirbelsäulenerkrankungen



Die Prozessfragmente wurden als Benutzeroberflächen implementiert, die bereits im Rahmen des SPOT-Projektes erstellt worden sind. Damit die Prozessfragmente vom Aristaflow® Server ausgeführt werden können, müssen sie zuvor im Repository verfügbar gemacht werden.

7 Zusammenfassung & Diskussion

Prozessmanagement gilt in vielen Domänen als die Lösung, um einerseits Maßnahmen zur Qualitätssicherung und Qualitätskontrolle durchzusetzen und andererseits Kosten einzusparen. Angesichts der demographischen Entwicklung und der steigenden Ausgaben im Gesundheitswesen, ergibt sich auch hier immer stärker der Bedarf nach prozessorientierten Systemlösungen. Da Krankenhäuser mit der Einführung des Fallpauschalensystems besonders unter Druck geraten sind, ihre Wettbewerbsfähigkeit zu erhalten, beginnen viele Häuser mit der Entwicklung und Anwendung klinischer Pfade. Die meisten Pfade existieren zunächst ausschließlich in narrativer oder tabellarischer Form und dienen dem Personal als einrichtungsinterner Leitfaden bei der Patientenbehandlung. Mit dem zunehmenden Einsatz von IT-Systemen in Krankenhäusern, z. B. zur Auftragskommunikation, Bettenbelegungssteuerung, Dokumentation und Abrechnung, stellt sich allerdings mehr und mehr die Anforderung, Konzepte und Werkzeuge bereitzustellen, die der elektronischen Repräsentation, Interpretation und Ausführung klinischer Pfade dienen.

Dabei liegt der Einsatz von WfMS nahe, deren Entwicklung bereits in den 90er Jahren begonnen hat und die in vielen anderen Domänen erfolgreich genutzt werden, um Prozesse zu modellieren, zu simulieren, auszuführen und kontinuierlich weiterzuentwickeln. Bei der Übertragung der WfM-Technologie auf den klinischen Bereich ergeben sich aber zahlreiche Probleme: So handelt es sich bei Behandlungsabläufen um sehr komplexe Prozesse, was die übersichtliche Darstellung in konventionellen Modellierungswerkzeugen enorm erschwert. Trotz Standardisierungsbemühungen durch klinische Pfade lässt sich die medizinische Versorgung von Patientinnen und Patienten nicht komplett vereinheitlichen; der hohe Anteil an Flexibilität in Folge des Auftretens von Varianz von klinischen Pfaden erschwert die Modellierungsarbeit zusätzlich. Neben traditionellen WfMS, die lediglich die Ausführung vollständig planbarer, starrer Prozessdefinitionen unterstützen, sind in den letzten Jahren flexible und sogar adaptive WfMS entstanden. Die WfMS dieser neuen Generation unterscheiden sich vor allem in einem grundsätzlichen Punkt: Dem Prozessmodellierungsparadigma. Prozedurale Modellierungssprachen erzwingen die exakte Festlegung, welche Aktivitäten, in welcher Reihenfolge auszuführen sind, und wann Datenobjekte produziert und konsumiert werden. Das deklarative Paradigma hingegen lässt die Reihenfolge, in der die Prozessaktivitäten bearbeitet werden, bis zur Instanziierung und Laufzeit des Prozesses weitgehend offen; die Anwender können flexibel entscheiden, wann sie welche Aufgaben durchführen möchten. Obwohl der deklarative Ansatz die Flexibilität bei der Prozessausführung deutlich erhöht und strukturelle Anpassungen zur Laufzeit unnötig machen soll, ist er mittelfristig nur bedingt zur Umsetzung klinischer Pfade geeignet. Grund dafür ist, dass die Bedeutung von IT-Systemen im Gesundheitswesen in den kommenden Jahren weiter steigen wird und zunehmend Aufgaben, die bisher noch manuell erledigt werden, automatisiert ablaufen; dazu zählen z. B. Terminierungsvorgänge oder die Buchung von Ressourcen. Je genauer die Aussagen der Prozessdefinition im Hinblick auf die Ablaufreihenfolge der Prozessaktivitäten sind, desto besser kann das IT-System diese Aufgaben erfüllen. Die dafür notwendige exakte Prozessplanung ist auf Basis deklarativer Modelle allerdings nicht möglich. Je mehr die Automatisierung administrativer Arbeiten im Gesundheitswesen zunimmt, desto höher wird auch der prozedurale Anteil innerhalb von Prozessdefinitionen.

Bei der Realisierung klinischer Pfade in Krankenhausinformationssystemen hat sich bisher das deklarative Paradigma weitgehend durchgesetzt. Dies ist zum einen der Erkenntnis ge-

schuldet, dass es für Fachexperten ohne tiefgreifende technische Kenntnisse meist nicht möglich ist, ausführbare prozedurale Prozessdefinitionen zu erstellen; zum anderen erleichtern deklarative Sprachen den Umgang mit Varianz von klinischen Pfaden. Ausgehend davon sind Produkte entstanden, die es Fachanwendern zwar ermöglichen, selbst deklarative Prozessdefinitionen zu entwerfen, die jedoch auch tief in die jeweiligen Systeme integriert und daher nicht auf andere Plattformen übertragbar sind. Darüber hinaus verfügen sie über keine Formalismen, die die korrekte Komposition von Prozessteilen zu vollständigen klinischen Pfaden sicherstellen. Auch der schrittweise Übergang zu prozeduralen Prozessdefinitionen, die immer noch dynamisch an eine individuelle Behandlungssituation angepasst werden können, wird durch diese Systeme nicht unterstützt.

Das übergeordnete Ziel dieser Arbeit bestand also in der Entwicklung eines Lösungskonzeptes, das Fachexperten in die Lage versetzt, selbst ausführbare und möglichst prozedurale Prozessdefinitionen zu erstellen und diese bei Bedarf flexibel zu verändern. Um dieses Ziel zu erreichen, wurde mit SPF-Typ-Graphen, deren Konfiguration, der Transformation der SPF-Graphen und ihrer optionalen Rekonfiguration ein Konzept eingeführt, das Anwendern die Prozessmodellierung und –anpassung ermöglicht, ohne sich mit technologischen Details, unterschiedlichen Schnittstellen und Modellierungsparadigmen auseinandersetzen zu müssen. Welcher Grad an Prozessflexibilisierung erreicht werden kann, hängt jedoch weiterhin von der zugrunde liegenden Technologie ab. Die Erstellung von SPF-Typ-Graphen, deren Graphgrammatik-basierte Konfiguration sowie die Rekonfiguration zur Modellierungszeit finden unabhängig von den eingesetzten WfMS statt. Bei einer vollständigen Konfiguration kann der resultierende SPF-Graph in beliebige Prozessmodellierungssprachen transformiert werden; anschließend ist die Prozessausführung auch mit einem starren WfMS möglich. Im Gegensatz zur Transformation und Ausführung vollständiger Konfigurationen müssen im Hinblick auf unvollständige Konfigurationen bereits gewisse Flexibilisierungskonzepte auf technologischer Ebene umgesetzt worden sein. Minimal muss die Möglichkeit bestehen, den unvollständigen Teil des SPF-Graphen durch Nutzung bedingter Verweigungen in der Zielsprache adäquat auszudrücken; dies ist jedoch mit einigen Schwierigkeiten verbunden (siehe Kapitel 4.5.5). Wie in dem referenzierten Kapitel dargestellt, besteht eine bessere Lösung darin, Late Binding bzw. Late Composition-Ansätze zu nutzen, um die verbliebenen nicht-terminalen Knoten im SPF-Graphen noch nachträglich zur Laufzeit zu konfigurieren. Dies macht vor allem dann Sinn, wenn signalisiert werden soll, dass in bestimmten Regionen des Prozesses noch Entscheidungen ausstehen, die erst auf Basis von Laufzeitparametern getroffen werden können. Die Nutzung dieser Konzepte setzt allerdings voraus, dass keine strukturellen Änderungen an anderen Prozessteilen, wie sie z. B. in Folge von Constraint-Relationen entstehen können, notwendig sind; d. h. unterstützt ein WfMS lediglich solche Konzepte, muss bei der Auswahl der nachträglich zu konfigurierenden Subgraphen berücksichtigt werden, dass keine entsprechenden Constraint-Relationen in andere, bereits konfigurierte Subgraphen existieren. Solche Einschränkungen können letztlich nur dann ausgeräumt werden, wenn adaptive WfMS wie ADEPT2 als technologische Basis genutzt werden. Sie sind darüber hinaus die Voraussetzung für die Realisierung der flexiblen Rekonfiguration von SPF-Typ-Graphen zur Laufzeit und damit auch für den Einsatz von Prozessvarianten.

Ausgehend von der detaillierten Betrachtung klinischer Pfade und des Phänomens der Varianz wurden zahlreiche Anforderungen erhoben, die den übergeordneten Kategorien »Entwicklung und Ausführung klinischer Pfade«, »Unterstützung manueller Abweichungsmaßnahmen« und »Unterstützung von Prozessvarianten« zugeordnet wurden. Auf Basis des Anforderungskatalogs wurden in Kapitel 1 sowohl praktische Ansätze zur Realisierung klinischer Pfade in Krankenhäusern als auch wissenschaftliche Lösungen zum Umgang mit Abweichungen von

Standardprozessen untersucht. Nun wird das in dieser Arbeit vorgestellte Lösungskonzept im Hinblick auf die Erfüllung der aufgestellten Anforderungen bewertet. Einen Überblick über die Ergebnisse aus der ersten Kategorie zeigt Tabelle 34.

Tabelle 34 Ergebnisse der Analyse des Lösungskonzeptes zur Entwicklung und Ausführung klinischer Pfade

Zentrale Anforderung	Resultierende Teilanforderung		Lösungskonzept
1. Entwicklung und Ausführung klinischer Pfade (siehe Kapitel 2.1.5)	1.1	Klinische Pfade müssen für Fachanwender modellierbar sein	**Ja** (Abbildung auf Fachsichten erforderlich)
	1.2	Optimierung des Grades an Wiederverwendbarkeit durch formale Methoden zur Verstetigung des Prozesswissens	**Ja** (Nutzung formaler SPF-Typ-Graphen)
	1.3	Semantische Abhängigkeiten zwischen Prozessteilen müssen formalisiert bei der Modellierung berücksichtigt werden	**Ja** (Grammatikbasierte Konfiguration, automatische Prüfung von Constraints)
	1.4	Die Modellierung unterschiedlicher Prozessteile kann gleichzeitig und gleichartig erfolgen	**Ja** (Gemeinsame Konfiguration von Klonknoten)
	1.5	Die Ausführung klinischer Pfade nach der Modellierung muss mit unterschiedlichen WfMS möglich sein	**Ja** (Regelbasierte Transformation)
	1.6	Es besteht die Möglichkeit, klinische Pfade in unterschiedliche Darstellungsformen zu überführen	**Ja** (Regelbasierte Transformation, Sichten auf SPF-Graphen)

Teilanforderung 1.1:

Auch wenn der im Rahmen der prototypischen Implementierung entwickelte SPOT-Domain-Model-Configurator nicht für den praktischen Einsatz in Kliniken geeignet ist, wurde mit dem SPOT-Care-Plan-Modeler bereits vorab ein Werkzeug geschaffen, das die Prozessmodellierung speziell für Anwender aus dem Gesundheitsbereich ermöglichen soll. Auf Basis des SPF-Typ-Graphen und der vom SPOT-Domain-Model-Configurator erzeugten Graphgrammatik ist es ohne großen Aufwand möglich, ansprechende Darstellungen im SPOT-Care-Plan-Modeler zu erzeugen. Der dort dargestellte initiale Behandlungsplan ergibt sich durch die automatische Konfiguration nach Auswahl des SPF-Typ-Graphen; die zur Verfügung stehenden Produktionsregeln ermöglichen anschließend die Konfiguration des Behandlungsplanes. Da der SPOT-Care-Plan-Modeler zwischen Behandlungsabschnitten und Behandlungsschritten differenziert, muss es im SPF-Typ-Graphen ein Attribut geben, das in jedem Teilzweig genau einen Kontextknoten als Behandlungsabschnitt markiert. Die Behandlungsschritte entsprechen ausschließlich den SPF Knoten, die dem markierten Kontextknoten als Nachfolger zugeordnet sind. Aufgrund der ohnehin kompositionalen Modellierungsmethodik bei den kommerziellen Werkzeugen zur Erstellung klinischer Pfade mit iMedOne® und ClinPath lässt sich das Lösungskonzept dort ebenfalls als formale Basis integrieren. Die kalenderartige Darstellung klinischer Pfade in iMedOne® entspricht dann lediglich einer Sicht auf die Konfiguration eines SPF-Typ-Graphen, nämlich den aktuellen SPF-Graphen. Durch diese Konzepte schafft das Lösungskonzept alle Voraussetzungen, damit die Entwicklung klinischer Pfade für Fachanwender möglich ist, ohne sich auf eine spezifische Repräsentationsform festzulegen. Darüber

hinaus wird nach dem Paradigma »Correctness by Construction« durch formale Methoden sichergestellt, dass Fachanwender nur valide Konfigurationen erzeugen können. Durch die Generierung von Graphgrammatiken auf Basis von SPF-Typ-Graphen wird gewährleistet, dass die Regeln für die Konfiguration des Graphen und damit die Erstellung eines klinischen Pfades klar definiert sind; Fachanwender können keine Konfigurationen erstellen, die im Widerspruch zu den Regeln der Graphgrammatik stehen. Abhängigkeiten und Interrelationen zwischen einzelnen Prozessfragmenten wird durch die Spezifikation von Constraints Rechnung getragen, deren Einhaltung durch automatische Verfahren garantiert ist.

Teilanforderung 1.2:

Mit dem SPF-Typ-Graphen wurde ein Modell entwickelt, das die Verstetigung des Prozesswissens auf Basis formaler Methoden erlaubt. Durch definierte Korrektheitskriterien wird sichergestellt, dass Änderungen am SPF-Typ-Graphen nicht dazu führen, dass der Graph anschließend nicht mehr konfigurierbar ist.

Teilanforderung 1.3:

Semantische Abhängigkeiten zwischen Prozessteilen lassen sich zum einen durch die hierarchische Struktur des SPF-Typ-Graphen verbunden mit der Angabe logischer Operatoren und zum anderen durch die Spezifikation von Constraint-Relationen festlegen. Durch die formale Definition von SPF-Typ-Graphen, ihren Korrektheitskriterien und der Spezifikation von Algorithmen zur automatischen Einhaltung von Constraints, wird die Beachtung der semantischen Abhängigkeiten erzwungen. Darüber hinaus wurde aufgezeigt, welche Schritte notwendig sind, um SPF-Typ-Graphen um weitere Constraint-Relationen zu ergänzen.

Teilanforderung 1.4:

Neue IT-Anwendungen lassen sich im Gesundheitswesen nur durchsetzen, wenn sie den zusätzlichen Aufwand für Fachanwender gering halten. Im Hinblick auf klinische Pfade bedeutet dies, dass ihre Modellierung möglichst effizient vom System unterstützt werden muss. Zu diesem Zweck wurden in dieser Arbeit Mechanismen eingeführt, die die gleichartige und gleichzeitige Konfiguration von Klonknoten im SPF-Graphen ermöglichen. Die Voraussetzung dafür ist, dass die Klonknoten über ein identisches Abbild im SPF-Typ-Graphen verfügen und dass für sie dieselbe Knotenkardinalität gewählt wurde.

Teilanforderung 1.5:

Anders als bisher existierende kommerzielle Werkzeuge für die Erstellung und Ausführung klinischer Pfade, ist das hier vorgestellte Lösungskonzept nicht auf bestimmte Modellierungssprachen oder Ausführungsumgebungen festgelegt. Die Definitions- und Konfigurationsphasen können vollkommen systemunabhängig realisiert werden. Erst im Kontext der Transformationsphase muss die Wahl einer definierten Modellierungssprache und eines WfMS, das die Ausführung der klinischen Pfade übernehmen soll, erfolgen. Sofern es sich bei den mit den SPF Knoten verknüpften Prozessfragmenten um ausführbare Anwendungskomponenten handelt und bei der Spezifikation des SPF-Typ-Graphen Datenabhängigkeiten zwischen den Fragmenten berücksichtigt wurden, ist der klinische Pfad sofort nach der Transformation ausführbar.

Teilanforderung 1.6:

Mit der Transformationsphase wurde im Lösungskonzept ein Abschnitt definiert, der speziell der Überführung der SPF-Graphen klinischer Pfade in unterschiedliche Darstellungsformen gewidmet ist. Dabei kann es sich sowohl um Fachsprachen als auch um technische Repräsentationsformen handeln. In dieser Arbeit wurde die Umwandlung von SPF-Graphen am Beispiel von ADEPT2 WSM Nets beschrieben.

Die nächste Tabelle stellt die Analyseergebnisse im Hinblick auf die Unterstützung manueller Abweichungen von klinischen Pfaden dar.

Tabelle 35 Ergebnisse der Analyse des Lösungskonzeptes zur Unterstützung manueller Abweichungsmaßnahmen

Zentrale Anforderung	Resultierende Teilanforderung		Lösungskonzept
2. Unterstützung manueller Abweichungsmaßnahmen	2.1	Manuelle Abweichungsmaßnahmen vom klinischen Pfad müssen für Fachanwender durchführbar sein (siehe Kapitel 2.2.3.1, 2.2.3.4.1, 2.2.3.4.4)	**Ja** (Grammatikbasierte Rekonfiguration)
	2.2	Fachanwender und WfMS müssen klar zwischen Standardprozess und vom Standard abweichenden Maßnahmen unterscheiden können (siehe Kapitel 2.2.3.1, 2.2.3.2)	**Ja** (Unterschiede in der Gestaltung der SPF-Graphen)
	2.3	Die Wahl der Abweichungsmaßnahmen muss in Abhängigkeit des Behandlungsfalls variieren können (siehe Kapitel 2.2.3.3)	**Ja** (beliebig, sofern Konformanz zum SPF-Typ-Graphen gegeben ist)
	2.4	Fachexperten müssen sowohl die Möglichkeit haben sofort als auch verzögert auf eine Varianz zu reagieren (siehe Kapitel 2.2.3.4.2)	**Ja** (Rekonfiguration ist jederzeit möglich)
	2.5	Abweichungsmaßnahmen können zu unterschiedlichen Zeitpunkten und an unterschiedlichen Stellen im Prozess wiederholt werden (siehe Kapitel 2.2.3.4.3)	**Ja** (lediglich die Prozesshistorie von Instanzen darf nicht geändert werden)
	2.6	Dieselben Abweichungsmaßnahmen lassen sich gleichzeitig an mehreren Stellen des Prozesses durchführen (siehe Kapitel 2.2.3.4.5)	**Ja** (Gemeinsame Konfiguration von Klonknoten)
	2.7	Semantische Abhängigkeiten zwischen Abweichungsmaßnahmen müssen bei ihrer Auswahl berücksichtigt werden (siehe Kapitel 2.2.3.4.6)	**Ja** (Grammatikbasierte Konfiguration, automatische Prüfung von Constraints)
	2.8	Dieselben Abweichungsmaßnahmen können unabhängig von bestimmten klinischen Pfaden durchgeführt werden (siehe Kapitel 2.2.3.4.7)	**Ja** (sofern klinische Pfade von demselben SPF-Typ-Graphen abgeleitet sind)
	2.9	Die Ausführung des klinischen Pfades muss auch nach der Durchführung manueller Abweichungsmaßnahmen möglich sein (siehe Kapitel 2.1.5)	**Ja** (Regelbasierte Transformation nach der Rekonfiguration)

Teilanforderung 2.1:

Da sich die Durchführung manueller Abweichungsmaßnahmen kaum von der Ausführung normaler Konfigurationsmaßnahmen unterscheidet, können Fachanwender klinische Pfade anpassen, wenn sie auch in der Lage waren, die Pfade selbst zu erstellen. In Ergänzung zur normalen Konfiguration müssen lediglich die zu rekonfigurierenden Knoten bestimmt werden. Vorschläge zu möglichen strukturerhaltenden Rekonfigurationen können unter Nutzung der in Kapitel 4.6 definierten Vorbedingungen vom System generiert werden; dabei sollte sich die Präsentation dieser Vorschläge für den Anwender nicht von der Darstellung der normalen Konfigurationsoptionen unterscheiden. Fachanwender können sich jedoch auch zur nicht-strukturerhaltenden Rekonfiguration entschließen. In diesem Fall wird das terminale Symbol des Knoten auf sein nicht-terminales Symbol zurückgesetzt; anschließend kann der Knoten erneut wie bei der initialen Pfadmodellierung konfiguriert werden. Bei Prozessinstanzen können die Optionen zur Rekonfiguration von Knoten im SPF-Graphen eingeschränkt sein; dieses bedeutet für Fachanwender jedoch lediglich, dass die entsprechenden Optionen nicht zur Auswahl stehen. Insgesamt gibt der SPF-Typ-Graph vor, welche Abweichungsmaßnahmen möglich sind. Je nach Gestaltung des Graphen kann die Maßnahmenergreifung abhängig vom Beobachtungszeitpunkt stark variieren. Auch die Maßnahmen selbst können je nach Effekt der Varianz unterschiedlich sein. Maßnahmen, die im SPF-Typ-Graphen generell nicht vorgesehen sind, können allerdings im klinischen Pfad auch nicht durchgeführt werden. Es besteht jedoch die Option, unterhalb beliebiger Kontextknoten SPF-Knoten einzufügen, die mit einem Prozessfragment verknüpft sind, das zumindest die Dokumentation dieser ungeplanten Maßnahmen ermöglicht. Darüber hinaus könnten Konzepte erarbeitet werden, die die spontane Weiterentwicklung von SPF-Typ-Graphen und die Evolution ihrer SPF-Graphen (siehe Kapitel 8.4) im Sinne eines Late Modeling Ansatzes (siehe Kapitel 3.2.3) unterstützen; dies kann durch Fachanwender ohne technische Kenntnisse aktuell allerdings kaum geleistet werden.

Teilanforderung 2.2:

Jeder klinische Pfad basiert auf einem SPF-Graphen als Konfiguration eines SPF-Typ-Graphen. Abweichungen entstehen durch die Modifikation des SPF-Graphen und seine erneute Transformation in einen ausführbaren Prozess und gegebenenfalls in eine Fachsicht auf diesen Prozess. Änderungen am SPF-Graphen werden sowohl vom System registriert als auch vom Fachanwender, der sich aktiv für die Durchführung einer solchen Anpassung entscheidet. Alle vom Standard abweichenden Maßnahmen sind also als Differenz zwischen dem ursprünglichen SPF-Graphen des klinischen Pfades und dem modifizierten SPF-Graphen feststellbar. Dies gilt sowohl bei Rekonfigurationen am SPF-Graphen des klinischen Pfades als auch bei der Ableitung individueller Prozessdefinitionen aus demselben SPF-Typ-Graphen; da jede Konfiguration einen Graphmorphismus auf den SPF-Typ-Graphen definieren muss, sind sämtliche Prozessdefinitionen miteinander vergleichbar.

Teilanforderung 2.3:

Die Wahl der Abweichungsmaßnahmen vom klinischen Pfad kann von Fall zu Fall unterschiedlich sein und ergibt sich durch die Selektion verschiedener Rekonfigurationsoperationen oder durch die Erstellung vollkommen neuer SPF-Graphen, die sich von dem SPF-Graphen des klinischen Pfades abheben. Welche Abweichungsmaßnahmen zur Verfügung stehen, wird allerdings durch den SPF-Typ-Graphen und die aus ihm generierte Graphgrammatik determiniert. Auf diese Weise wird sichergestellt, dass der für einen Patienten individualisierte klinische Pfad stets mit den grundsätzlichen Regulationen einer Einrichtung korrespondiert.

Teilanforderung 2.4:

Das Lösungskonzept stellt prinzipiell keine einschränkenden Forderungen auf, die die Wahl von Abweichungsmaßnahmen auf bestimmte Zeitpunkte oder Bereiche des klinischen Pfades beschränken. Das Einfügen neuer Knoten in einen SPF-Graphen ist grundsätzlich immer möglich, sofern der Graph anschließend noch den Vorgaben aus dem SPF-Typ-Graphen entspricht; dies gilt für Prozessinstanzen ebenso wie für Prozessdefinitionen. Lediglich sobald Knoten aus dem SPF-Graphen gelöscht werden sollen, können sich für Prozessinstanzen Einschränkungen ergeben, da das Löschen bereits ausgeführter Prozessaktivitäten nicht erlaubt ist.

Teilanforderung 2.5:

Prinzipiell können dieselben Abweichungsmaßnahmen zu beliebigen Zeiten und an verschiedenen Stellen des klinischen Pfades durchgeführt werden. Dies erhöht in keiner Weise die Komplexität der zugrunde liegenden Prozessdefinitionen. Potentielle Einschränkungen ergeben sich auch hier ausschließlich im Hinblick auf die nachträgliche Änderung der Prozesshistorie.

Teilanforderung 2.6:

Durch die Möglichkeit zur gemeinsamen Rekonfiguration und Konfiguration einer Menge von Knoten können Abweichungsmaßnahmen auch gleichzeitig an unterschiedlichen Bereichen vorgenommen werden. Die Auswahl der entsprechenden Knoten kann entweder manuell durch Fachanwender erfolgen oder mit Hilfe des in Kapitel 4.6.3 spezifizierten Identifikationsalgorithmus.

Teilanforderung 2.7:

Wie bei der Konfiguration so wurden auch für die Rekonfiguration Methoden definiert, die sicherstellen, dass die im SPF-Typ-Graphen aufgestellten semantischen Abhängigkeiten bei der Individualisierung klinischer Pfade berücksichtigt werden. Anders als bei der Konfiguration und Rekonfiguration zur Modellierungszeit wird zur Laufzeit die Einhaltung der Constraint-Relationen nicht automatisch erzwungen. Vielmehr bleibt eine Rekonfigurationsoperation untersagt, so lange sie gegen Constraints verstößt. Auf diese Weise wird das Auftreten kaskardierender Änderungen, mit gegebenenfalls unerwünschten Effekten für den laufenden Behandlungsprozess, unterbunden.

Teilanforderung 2.8:

Sofern eine Menge von klinischen Pfaden auf demselben SPF-Typ-Graphen beruht, d. h. einen entsprechenden Graphmorphismus definiert, können an diesen Pfaden dieselben Änderungsmaßnahmen durchgeführt werden. Auf diese Weise ist die Wahl der Maßnahmen unabhängig von spezifischen Prozessen. Die Notwendigkeit zur mehrfachen und aufwändigen Spezifikation von Änderungsmaßnahmen (z. B. durch die Einbettung bedingter Verzweigungen) oder der Stellen, an denen die Änderungsmaßnahmen ansetzen sollen (z. B. durch Variationspunkte in Prozessdefinitionen) wird so vermieden.

Teilanforderung 2.9:

Wird bei der Rekonfiguration und erneuten Transformation des SPF-Graphen die Ausführungssemantik des gewählten WfMS berücksichtigt, können Abweichungsmaßnahmen an klinischen Pfaden durchgeführt werden, ohne die weitere Ausführbarkeit der Prozessdefinition bzw. der Prozessinstanz zu gefährden.

Zuletzt soll analysiert werden, ob das Lösungskonzept auch die mit der Umsetzung von Prozessvarianten verbundenen Anforderungen abdecken kann (siehe Tabelle 36).

Tabelle 36 Ergebnisse der Analyse des Lösungskonzeptes zur Unterstützung von Prozessvarianten

Zentrale Anforderung	**Resultierende Teilanforderung**		**Lösungskonzept**
3. Unterstützung von Prozessvarianten	3.1	Prozessvarianten als standardisierte Bündel von Abweichungsmaßnahmen müssen für Fachanwender modellierbar sein (siehe Kapitel 2.2.3.4.1, 2.2.3.4.4)	**Ja** (Konfiguration von SPF-Typ-Graphen für Varianten)
	3.2	Fachanwender und WfMS müssen klar zwischen Standardprozess und Variante unterscheiden können (siehe Kapitel 2.2.3.1, 2.2.3.2)	**Ja** (Unterschiedliche Gestaltung der SPF-Graphen)
	3.3	Semantische Abhängigkeiten zwischen einzelnen Abweichungsmaßnahmen einer Prozessvariante müssen berücksichtigt werden (siehe Kapitel 2.2.3.4.6)	**Ja** (Grammatikbasierte Konfiguration, automatische Überprüfung von Constraints)
	3.4	Fachanwender müssen Prozessvarianten in Abhängigkeit des Behandlungsfalls anpassen können (siehe Kapitel 2.2.3.4.1, 2.2.3.4.4)	**Ja** (Grammatikbasierte Rekonfiguration)
	3.5	Prozessvarianten können unabhängig von bestimmten klinischen Pfaden definiert und eingesetzt werden (siehe Kapitel 2.2.3.4.7)	**Ja** (Prozessvarianten müssen sich auf den SPF-Typ-Graphen beziehen, von dem die Pfade abgeleitet wurden)
	3.6	Konflikte mit manuellen Abweichungsmaßnahmen werden vermieden (siehe Kapitel 2.2.3.4.1)	**Ja** (SPF-Typ-Graph als gemeinsame Basis aller Änderungen)
	3.7	Die Ausführung des klinischen Pfades muss auch nach der Auswahl einer Prozessvariante möglich sein (siehe Kapitel 2.1.5)	**Ja** (Regelbasierte Transformation nach der Rekonfiguration)

Teilanforderung 3.1:

Die Entwicklung von Prozessvarianten folgt exakt demselben Vorgehen wie die Erstellung klinischer Pfade. D. h. IT-Spezialisten werden mit der Erstellung der notwendigen Prozessfragmente betraut; diese bewirken den Aufruf von SPF-Änderungsoperationen mit jeweils unterschiedlichen Parametern. Hierzu genügt z. B. die Umsetzung eines einzelnen Web Service, der als Methoden die verschiedenen Änderungsoperationen bereitstellt. Die statischen Parameterwerte können bereits in den Schnittstellenspezifikationen der Prozessfragmente angegeben werden; die Ermittlung der Werte für die dynamischen Parameter muss durch Interakti-

on mit dem Benutzer im Rahmen des Top-Level-Prozesses für Prozessvarianten stattfinden. Nach der Entwicklung von SPF-Typ-Graphen für einzelne oder Gruppen von Prozessvarianten und der Verknüpfung der SPF Knoten mit den erstellten Fragmenten, können die Prozessdefinitionen für die Varianten von Fachanwendern selbständig durch Konfiguration abgeleitet werden. Auf diese Weise kann der Umgang mit Varianz standardisiert im Bedarfsfall aber auch variiert werden.

Teilanforderung 3.2:

Da die Anwendung einer Prozessvariante genau wie manuelle Abweichungsmaßnahmen zu Modifikationen am SPF-Graphen des klinischen Pfades führen, entspricht die Differenz zwischen dem ursprünglichen und dem neuen Graphen dem Ergebnis der Anpassung. Auf dieser Basis ist die Unterscheidung zwischen Standard und Abweichung sowohl für die Fachanwender als auch für das WfMS möglich.

Teilanforderung 3.3:

Durch die Typisierung von SPF-Graphen und die automatische Kontrolle der Einhaltung von Constraints können auch mit Prozessvarianten nur solche Änderungen durchgeführt werden, die innerhalb des vom SPF-Typ-Graphen aufgespannten Möglichkeitsraum vorgesehen sind. Steht eine im Kontext einer Variante ausgeführte Änderungsoperation nicht in Konformanz zu den semantischen Abhängigkeiten zwischen den Prozessfragmenten, so wird diese Operation nicht ausgeführt und ihr Aufruf bleibt wirkungslos.

Teilanforderung 3.4:

Genau wie SPF-Graphen für klinische Pfade lassen sich auch SPF-Graphen für Prozessvarianten unterschiedlich (re-)konfigurieren. Je nachdem welche Möglichkeiten der SPF-Typ-Graph einräumt, kann ein Fachanwender also durch Rekonfiguration die Struktur einer Prozessvariante und die von ihr vorgesehenen Änderungsoperationen modifizieren.

Teilanforderung 3.5:

Jede Prozessvariante bezieht sich stets auf einen SPF-Typ-Graphen. Damit kann die Variante auf alle Prozessdefinitionen und –instanzen von denjenigen klinischen Pfaden angewendet werden, deren SPF-Graph auf diesem SPF-Typ-Graphen beruht.

Teilanforderung 3.6:

Beim Einfügen von Knoten in einen SPF-Graphen können Konflikte zwischen Prozessvarianten und manuellen Abweichungsmaßnahmen prinzipiell ausgeschlossen werden. Das Hinzufügen von Knoten ist nämlich immer möglich, sofern noch nicht die maximale Knotenkardinalität ausgeschöpft ist. Wurde diese erreicht, bleibt die Anwendung der Prozessvariante an dieser Stelle wirkungslos und das Einfügen wird nicht mehr als Option zur manuellen Anpassung angeboten. Beim Löschen können Wechselwirkungen entstehen, die jedoch logisch sind und nicht als Konflikt betrachtet werden können. Hat ein Fachanwender z. B. eine weitere MRT-Untersuchung in den SPF-Graphen eingefügt und wählt als nächstes die Prozessvariante zur Anpassung des klinischen Pfades für Patienten mit Herzschrittmacher, sollte die Variante dazu führen, dass alle MRT-Untersuchungen, einschließlich der manuell eingefügten, durch andere Bildgebende Verfahren ersetzt werden.

Teilanforderung 3.7:

Da die Prozessvariante analog zu manuellen Abweichungsmaßnahmen nur Änderungen im Rahmen der vom SPF-Typ-Graphen definierten Möglichkeiten gestattet, ist die Prozessdefinition bzw. die Instanz des klinischen Pfades auch nach der Anwendung einer Prozessvariante weiterhin ausführbar. Voraussetzung dafür ist, dass der SPF-Typ-Graph korrekt definiert wurde und alle Datenabhängigkeiten durch Spezifikation von Constraint-Relationen berücksichtigt wurden.

8 Ausblick

Obwohl mit dem in dieser Dissertation entwickelten und vorgestellten Lösungskonzept bereits ein umfassendes Rahmenwerk zur Modellierung und dynamischen Adaption klinischer Pfade auf Basis semantischer Prozessfragmente geschaffen wurde, gibt es immer noch einzelne Aspekte, denen nicht ausreichend Aufmerksamkeit gewidmet werden konnte, die aber im Rahmen zukünftiger Forschungsarbeiten angegangen werden können. Im Folgenden werden diese Themen jeweils kurz angerissen.

8.1 Validierung des Lösungskonzeptes im Kontext einer praktischen Fallstudie

Die fragmentorientierten, deklarativen Modellierungsansätze für klinische Pfade in Krankenhäusern und die weite Verbreitung von Konzepten wie Feature-Modellierung legen die Anwendbarkeit und Nutzerfreundlichkeit der in dieser Arbeit beschriebenen Mechanismen nahe. Darüber hinaus wurde mit der Entwicklung der prototypischen Implementierung ein Nachweis für die Umsetzbarkeit der grundlegenden, formalen Konzepte erbracht. Die Validierung des Lösungskonzeptes fand in Kapitel 1 an Hand eines einfachen Prozessbeispiels statt; dieses Beispiel ist während eines Projektes im Auftrag der Krankenversicherung BIG Direkt entstanden und der Prozess wurde in Abstimmung mit einem Neurochirurgen entwickelt. Dennoch bleiben vor allem im Hinblick auf die Gestaltung von SPF-Typ-Graphen noch einige Fragen offen, die nur im Kontext einer groß angelegten Fallstudie beantwortet werden können. Einige Fragen beziehen sich z. B. auf die zunehmende Komplexität des Graphen, wenn immer mehr klinische Pfade darüber abgebildet werden sollen. Wie viele Aspekte eines Geschäftsprozesses oder einer Menge von Geschäftsprozessen lassen sich in einem SPF-Typ-Graphen integrieren und wann macht die Auslagerung von Teilen der Spezifikation in unterschiedliche SPF-Typ-Graphen Sinn? Ist es möglich, die Spezifikationsteile anschließend wieder in einem übergeordneten Modell zusammenzufassen? Ist dies sinnvoll und praktikabel oder ist der Unterteilung in viele kleinere Prozessabschnitte der Vorzug zu geben? Die Integration separater SPF-Typ-Graphen in einen gemeinsamen Graphen erfordert zudem Möglichkeiten der Komposition von SPF-Typ-Graphen analog zu Subprozessen, die in dieser Arbeit nicht behandelt wurden. Diese Aspekte haben auch Auswirkungen auf die Gestaltung von Prozessvarianten, da sich der SPF-Typ-Graph einer Variante immer auf den SPF-Typ-Graphen einer Menge von klinischen Pfaden beziehen muss. Eine Prozessvariante ist also umso universeller einsetzbar, je mehr klinische Pfade aus ein und demselben SPF-Typ-Graphen abgeleitet werden können.

Derzeit läuft am Fraunhofer ISST das Forschungsprojekt »Telemedizin-Repository«, dessen Ziel in der Bereitstellung von Methoden zur prozessorientierten Komposition telemedizinischer Lösungsbausteine zu kompletten Anwendungen unter Berücksichtigung von Sicherheitsaspekten und Standards besteht [Reuter et al. 2009, Reuter et al. 2010]. In dem Projekt kommen die in dieser Arbeit entwickelten Lösungen zur Spezifikation von Prozessfragmenten und zur Anordnung von Fragmenten in SPF-Typ-Graphen zum Einsatz. Dabei werden SPF-Typ-Graphen jedoch noch um Informationen zu Sicherheit und Standards ergänzt. SPF-Typ-Graphen werden nicht nur als Basis für die Komposition von Prozessfragmenten zu ausführbaren Prozessdefinitionen verwendet, sondern auch zur Konfiguration ausführbarer Softwareapplikationen. Die Ergebnisse dieses Projektes werden daher wichtige Anhaltspunkte liefern, um die oben genannten Fragen zu beantworten und die Praktikabilität des Lösungskonzeptes in komplexen Szenarien zu untersuchen.

8.2 Explizite Berücksichtigung weiterer Workflow-Aspekte bei der Konfiguration und Rekonfiguration von SPF-Typ-Graphen

SPF-Typ-Graphen bilden derzeit lediglich den funktionalen und den verhaltensorientierten Workflow-Aspekt ab. Der wichtige informationsbezogene Aspekt sowie die organisatorische und die operationale Perspektive sind hingegen implizit über die Schnittstellenspezifikationen der Prozessfragmente verfügbar. Hier ergibt sich die Frage, wie zusätzliche Workflow Aspekte direkt im SPF-Typ-Graphen dargestellt werden können, um Mechanismen zur Überprüfung der Korrektheit möglicher Konfigurationen z. B. im Hinblick auf Informationsflüsse effizient umsetzen zu können.

Im Projekt »Telemedizin-Repository« werden die Aspekte Sicherheit und Standards als zusätzliche Perspektiven ergänzt. Diese können Einfluss auf die Strukturierung von SPF-Typ-Graphen nehmen. Z. B. hängen die Verschlüsselungsanforderungen medizinischer Daten, die zwischen unterschiedlichen Einrichtungen ausgetauscht werden, u. a. von dem Speicherort der Daten ab. Werden die Daten in einer medizinischen Einrichtung aufbewahrt, genügt aufgrund des dort geltenden Beschlagnahmeschutzes die Transportverschlüsselung; ansonsten müssen die Daten selbst verschlüsselt werden. Im Hinblick auf den SPF-Typ-Graphen könnte sich also ein Kontextknoten auf den Speicherort der Daten beziehen und bei den nachfolgenden Knoten würde dann zwischen »Medizinischer Einrichtung« bzw. »Nicht-medizinische Einrichtung« als Lokation unterschieden. Je nach Auswahl würden anschließend andere Prozessfragmente für die Verschlüsselung zur Anwendung kommen. Für die genannten Workflow-Aspekte kann untersucht werden, ob ähnliche Verfahren genutzt werden können oder erweiterte Methoden spezifiziert werden müssen.

8.3 Unterstützung des Prozesswechsels

Bei der Nutzung klinischer Pfade spielt nicht nur die Spezifikation von Standardprozessen und die Bereitstellung von Möglichkeiten zur Abweichung von diesen Standards eine Rolle, sondern auch die Einleitung eines Prozesswechsels. In Kapitel 2.2 wurde die Kategorie der »Pfadabbrecher« vorgestellt. Dabei handelt es sich um Patientinnen oder Patienten, die bereits auf einen klinischen Pfad gesetzt worden sind. Noch vor Beendigung des Pfades wird allerdings festgestellt, dass die weitere Behandlung gemäß Pfad nicht möglich ist. Gründe dafür sind z. B., dass sich die Aufnahmediagnose nach der Durchführung diagnostischer Maßnahmen nicht bestätigt hat, oder dass die Behandlung einer anderen Diagnose eine höhere Priorität eingeräumt werden muss. Darüber hinaus kann es insbesondere aber bei schwer standardisierbaren Behandlungsabläufen, wie z. B. bei der internistischen Versorgung, vorkommen, dass strikt zwischen symptombezogenen, diagnostischen und therapeutischen Pfaden unterschieden wird. So besteht die Möglichkeit, nach Abschluss des symptombezogenen klinischen Pfades mit der Instanziierung des diagnostischen Pfades weiter zu verfahren usw. Dabei kann es jedoch vorkommen, dass die Diagnose bereits feststeht, bevor der symptombezogene Pfad abgeschlossen ist; in diesem Fall macht es keinen Sinn, den Prozess zum Abschluss zu bringen, der Pfad kann also abgebrochen werden, um sofort mit der Diagnostik oder Therapie zu beginnen.

Nach dem Abbruch eines klinischen Pfades und der Instanziierung eines neuen Pfades, kommt es häufig vor, dass im zweiten Pfad Untersuchungen oder Prozeduren vorgesehen sind, die bereits für diesen Behandlungsfall durchgeführt wurden. Um Doppeluntersuchungen zu vermeiden, muss also ein Abgleich zwischen den beiden Pfaden erfolgen. Maßnahmen, die

bereits im ersten Pfad stattfanden, müssen im zweiten gegebenenfalls nicht noch einmal wiederholt werden. Darüber hinaus besteht die Möglichkeit, dass der erste Pfad bereits modifiziert wurde; entweder unter Nutzung manueller Abweichungsmaßnahmen oder durch Auswahl von Prozessvarianten. Solche Änderungen sollten auch für den zweiten Pfad weiterhin berücksichtigt werden. Außerdem können Interrelationen zwischen Prozessfragmenten den Prozesswechsel erschweren, wenn z. B. im ersten Pfad Fragmente ausgeführt wurden, die in Ausschlussbeziehung zu bestimmten Fragmenten des zweiten Pfades stehen. Die Unterstützung des Prozesswechsels ist also nicht trivial, kann jedoch auf Basis des hier vorgestellten Lösungskonzeptes unterstützt werden, sofern beide Pfade auf demselben SPF-Typ-Graphen basieren. Die dafür erforderlichen Mechanismen und Algorithmen müssen jedoch noch spezifiziert werden.

8.4 Evolution von SPF-Typ-Graphen

Ein weiteres relevantes Thema ist die Evolution von SPF-Typ-Graphen für klinische Pfade und Prozessvarianten. Die Evolution ist zum einen erforderlich, um die kontinuierliche Weiterentwicklung des klinischen Pfades durch die Aufnahme neuer Behandlungsaspekte und Prozessfragmente zu ermöglichen und zum anderen, um alle laufenden Prozessinstanzen gemeinsam anzupassen, sofern sich grundlegende Änderungen an der Behandlungsplanung ergeben, die sofort durch die Instanzen umgesetzt werden müssen. Ein mögliches Szenario besteht z. B. darin, dass ein bestimmtes Medikament schwerwiegende Nebenwirkungen besitzt und nicht mehr verschrieben werden darf. Rinderle hat in ihrer Arbeit ein umfassendes Konzept zur Evolution von Prozessinstanzen erarbeitet [Rinderle et al. 2003, Rinderle et al. 2004, Rinderle 2004]. Die Anpassung der Instanzen auf Basis der geänderten Prozessdefinition wird erschwert, wenn bereits Änderungen an der Prozessinstanz durchgeführt worden sind und die Instanz also nicht mehr mit der ursprünglichen Prozessdefinition korrespondiert. In weiteren Forschungsarbeiten sollte überprüft werden, ob die in der Arbeit beschriebenen Methoden auch auf SPF-Typ-Graphen und die von ihnen abgeleiteten SPF-Graphen angewandt werden können. Bei der Prozessevolution kann z.B. der SPF-Typ-Graph so umstrukturiert werden, dass daraus entstandene SPF-Graphen nicht mehr konform zu ihm sind; dementsprechend müsste geprüft werden, ob die Möglichkeit besteht, den SPF-Graphen so zu verändern, dass die Konformanz zum SPF-Typ-Graphen wieder hergestellt wird. In einem weiteren Schritt müsste auch für die zum SPF-Graphen existierenden Prozessinstanzen analysiert werden, ob diese auf den angepassten SPF-Graphen migriert werden können.

Eine besondere Herausforderung besteht auch im Hinblick auf die Auswirkungen der Evolution klinischer Pfade auf die SPF-Typ-Graphen von Prozessvarianten. Werden z. B. Knoten aus dem SPF-Typ-Graphen entfernt, besteht die Möglichkeit, dass die Änderungsoperationen der Variante nicht mehr durchführbar sind. IT-Spezialisten müssen daher bei der Evolution von SPF-Typ-Graphen klinischer Pfade auch über die Auswirkungen auf Prozessvarianten unterrichtet und durch geeignete Methoden bei deren Anpassung unterstützt werden.

8.5 Multi-Level Prozessmanagement

Mit klinischen Pfaden als standardisierten Behandlungsprozessen für Patienten ausgehend von einem bestimmten Symptombild, einer Diagnose oder einer Prozedur wird bisher nur ein kleiner Teil der Prozesse, die in Krankenhäusern ablaufen, abgedeckt. Andere Prozesse beziehen sich auf organisatorische Maßnahmen, die mit der Nutzung von Ressourcen zusam-

menhängen, oder auf logistische Abläufe, z. B. zur Bestellung von Medikamenten. Die Instanzen der klinischen Pfade von Patienten sowie die organisatorischen und logistischen Prozesse stehen in vielseitigen Wechselbeziehungen zueinander. Eine Operation kann z. B. nur dann durchgeführt werden, wenn der Operationsraum desinfiziert wurde. Es ist jedoch nicht zielführend, die dafür notwendigen Aktivitäten in jeden klinischen Pfad zu integrieren; so könnte der Fall auftreten, dass der Raum durch die parallele Ausführung klinischer Pfade mehrmals hintereinander desinfiziert wird, ohne dass zwischendurch Operationen stattfinden, oder im kritischen Fall der OP vor der Durchführung der Prozedur nicht ordnungsgemäß gereinigt worden ist.

Effizienzsteigerungen bei der Buchung von Ressourcen oder bei der Durchführung logistischer Abläufe können nur dann erzielt werden, wenn die aktuell ablaufenden Prozesse optimal aufeinander abgestimmt und miteinander synchronisiert werden. Wenn bereits im Voraus bekannt ist, wann ein Untersuchungsraum für einen bestimmten Patient belegt wird, kann dies bei der Behandlungsplanung der anderen Fälle berücksichtigt werden und die Wartezeiten für Patienten lassen sich reduzieren.

Mit dem aktuellen Lösungskonzept werden durch die Definition von Constraints und logischen Operatoren in SPF-Typ-Graphen Abhängigkeiten und Beziehungen auf der strukturellen Ebene ein und derselben Prozessdefinition berücksichtigt. Die Betrachtung von Abhängigkeiten zwischen unterschiedlichen Prozessdefinitionen war nicht Gegenstand dieser Arbeit, wird jedoch in anderen Forschungsarbeiten behandelt [Ly et al. 2010]. Im Forschungsprojekt »Hospital Engineering« plant das Fraunhofer ISST die Interrelationen zwischen den in Krankenhäusern ablaufenden Prozessen sowie einrichtungsübergreifenden Prozessen mit anderen Leistungserbringern und Institutionen zu untersuchen. SPF-Typ-Graphen könnten den Rahmen bilden, um die Zusammenhänge zwischen Prozessfragmenten formal zu spezifizieren und bei der Ableitung spezifischer Prozessdefinitionen zu berücksichtigen. Indem SPF-Typ-Graphen ein übergeordnetes Modell über alle Prozessfragmente und deren Zusammenhänge und Abhängigkeiten beschreiben, könnten sie auch die Grundlage für ein effektives multi-level Prozessmanagement schaffen. Denn ein solches Modell bildet die Voraussetzung, um Abgleiche zwischen unterschiedlichen Prozessen durchzuführen und die Prozessabläufe effizient aufeinander abzustimmen.

9 Referenzen

Adams et al. 2005	M.J. Adams, A.H.M. ter Hofstede, D. Edmond, W.M.P. van der Aalst: Facilitating Flexibility and Dynamic Exception Handling in Workflows through Worklets. In O. Bello, J. Eder, O. Pastor, J. Falcao e Cunha (Hrsg.): Proceedings of the CAiSE'05 Forum, 45-50, Porto, Portugal, 2005
Adams et al. 2006	M. Adams, A.H.M. ter Hofstede, D. Edmond, W.M.P. van der Aalst: Worklets: A Service-Oriented Implementation of Dynamic Flexibility in Workflows. In R. Meersman, Z. Tari et al. (Hrsg.): Proceedings of the 14th Int'l Conference on Cooperative Information Systems (CoopIS'06), Lecture Notes in Computer Science, vol. 4275, 291-308, Springer Verlag Berlin / Heidelberg, Montpellier, France, 2006
Alberti et al. 1998	K.G.M.M. Alberti, P.Z. Zimmet: Definition, Diagnosis and Classification of Diabetes Mellitus and its Complications. Part 1: Diagnosis and Classification of Diabetes Mellitus. Provisional Report of a WHO Consultation, Diabetic Medicine, 15(7): 539-553, 1998
Andrews et al. 2003	T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, S. Weerawarana: Business Process Execution Language for Web Services, Version 1.1. Standards proposal by BEA Systems, International Business Machiens Corporation, and Microsoft Corporation, 2003
Andries et al. 1999	M. Andries, G. Engels, A. Habel, B. Hoffmann, H-J. Kreowski, S. Kuske, D. Plump, A. Schürr, G. Taentzer: Graph Transformation for Specification and Programming. Science of Computer Programming, Elsevier Nord-Holland Inc., 34(1): 1-54, 1999
Angstwurm et al. 1998	H. Angstwurm, K.D. Bachmann, R. Besser, D. Birnbacher, W.J. Bock, F.-W. Eigler, R.A. Frowein, G. Jorch, J. Reiter, O. Schober, H.-L. Schreiber, J. Schüttler, H.-B. Wuermeling: Richtlinien zur Feststellung des Hirntodes. Dritte Fortschreibung 1997 mit Ergänzungen gemäß Transplantationsgesetz (TPG). Deutsches Ärzteblatt 95, 30(53):A-1861-1868, 1998
Antkiewicz & Czarnecki 2004	M. Antkiewicz, K. Czarnecki: FeaturePlugin: Feature Modeling Plug-In for Eclipse. In Proceedings of the OOPSLA'04 Eclipse Technology eXchange (ETX) Workshop, 2004
Atwal & Caldwell	A. Atwal, K. Caldwell: Do multidisciplinary integrated care

2002	pathways improve interprofessional collaboration?, Scandinavian Journal of Caring Scienes, 2002, 16(4): 360-367
AWMF 2005	Dt. Ges. für Angiologie, Dt. Ges. für Chrirugie, und weitere AWMF Mitgliedsgesellschaften: Diagnostik und Therapie der Bein- und Beckenvenenthrombose und Lungenembolie, http://www.uni-duesseldorf.de/awmf/ll/065-002.htm, abgerufen am 13. Juli 2009
Bachmann & Bass 2001	F. Bachmann, L. Bass: Managing Variability in Software Architectures. In: Proc. of 2001 Symp. of Software Reusability, New Yor, ACM Press, 126-132, 2001
Barth Frink & Strassner 1996	B. Barth Frink, L. Strassner: Variance Analysis, In D. L. Flarey; S. Smith Blancett (Hrsg.): Handbook of Nursing Case Management, 1996, Aspen Publications, 194-223
Batory 2005	D. Batory: Feature Models, Grammars, and Propositional Formulas. In Proc. Int'l Software Product Line Conference, Vol. 3714 of LNCS, Springer Verlag, Heidelberg, 7-20, 2005
Bayer et al. 2005	J. Bayer, W. Buhl, C. Giese, T. Lehner, A Ocampo, F. Puhlmann, E. Richter, A. Schnieders, J. Weiland, M. Weske: PESOA - Process Family Engineering - Modeling variant-rich processes. Technischer Bericht 18/2005, Hasso-Plattner-Institut, Potsdam, 2005
Becker & Kahn 2005	J. Becker, D.Kahn: Der Prozess im Fokus. In: J. Becker, M. Kugeler, M. Rosemann (Hrsg.): Prozessmanagement. Ein Leitfaden zur prozessorientierten Organisationsgestaltung, 5. Auflage, S. 6
Becker et al. 2001	M. Becker, L. Geyer, A. Gilbert, K. Becker: Comprehensive Variability Modeling to Facilitate Efficient Variability Treatment. In: Proc. of the 4th Int'l. Workshop of Product Family Engineering, 2001
Becker et al. 2005	J. Becker, M. Kugeler, M. Rosemann (Hrsg.): Prozessmanagement. Ein Leitfaden zur prozessorientierten Organisationsgestaltung. 5. Auflage, Springer Berlin-Heidelberg, 2004
Bednasch 2002	T. Bednasch: Konzept und Implementierung eines konfigurierbaren Metamodells für die Merkmalmodellierung. Diplomarbeit, Fachbereich Informatik und Mikrosystemtechnik, Fachhochschule Kaiserslautern, Standort Zweibrücken, Deutschland, 2002
Benavides et al. 2005	D. Benavides, P. Trinidad, A. R. Cortés: Automated Reasoning on Feature Models. In O. Pastor, J. F. e Cunha (Hrsg.):

	CAiSE, Vol. 3520 of Lecture Notes in Computer Science, Springer Verlag, Heidelberg, 491-503, 2005
Benavides et al. 2006	D. Benavides, A. Ruiz-Cortés, P. Trinidad, S. Segura: A Survey on the automated analysis of feature models. Jornadas de Ingeniería del Software y Bases de Datos (JISBD'06), Barcelona, Spanien, 2006
Bentley 1986	J. L. Bentley: Programming pearls: Little languages. Communications of the ACM, 29(8): 711-721, 1986
Berge 1985	C. Berge: Graphs and Hypergraphs. Elsevier Science Ltd., 1985
Berge 1989	C. Berge: Hypergraphs: The Theory of Finite Sets. 1989
Bevan & Cawley 2006	H. Bevan, M. Cawley: Reducing costs and improving quality in the NHS, NHS Institute for Innovation and Improvement, 2006
Blockeel & Nijssen 2008	H. Blockeel, S. Nijssen: Induction of Node Label Controlled Graph Grammar Rules. Proc. of the 6th Int'l Workshop on Mining and Learning with Graph pages (MLG 2008), 1-4, 2008
BmG 2001	Bundesministerium für Gesundheit: Leistungsgerechte Vergütung durch diagnose-orientierte Fallpauschalen verbessert Qualität, Transparenz und Wirtschaftlichkeit in der stationären Versorgung - Informationen zum Gesetz zur Einführung des diagnose-orientierten Fallpauschalensystems für Krankenhäuser (Fallpauschalengesetz - FPG), 2001
BmG 2002	Bundesministerium für Gesundheit: Gesetz zur Einführung des diagnose-orientierten Fallpauschalensystems für Krankenhäuser (Fallpauschalengesetz - FPG), 2002
Bobrik 2008	R. Bobrik: Konfigurierbare Visualisierung komplexer Prozessmodelle. Dissertation, Universität Ulm, 2008
Bobrik et al. 2007	R. Bobrik, M. Reichert, T. Bauer: View-Based Process Visualization, In: 5th Int'l Conf. on Business Process Management (BPM'07), 2007, Brisbane, Australia
Böckmann 2010	B. Böckmann: IT-Unterstützung für transsektorale Behandlungspfade – Möglichkeiten und Grenzen unter Einbeziehung von Erfahrungen im Gesundheitsnetz Prosper. In: W. Hellmann, S. Eble (Hrsg.): Ambulante und Sektoren übergreifende Behandlungspfade – Konzepte Umsetzung Praxisbeispiele.Medizinisch Wissenschaftliche Verlagsgesellschaft, 197-

	209, 2010
Bowers 1995	K. Bowers: Case management along the continuum, Lecture sponsered by Contemporary Forums, 1995, San Francisco, Calif. Abstract
Breitbart et al. 1993	Y. Breitbart, A. Deacon, H.-J. Schek, A. Sheth, G. Weikum: Merging Application-centric and Data-centric Approaches to Support Transaction-oriented Multi-system Workflows, SIGMOND Record, Vol. 22, No. 3, September 1993, S. 23-30
Bretschneider & Bohnet-Joschko 2007	U. Bretschneider, S. Bohnet-Joschko: Prozessmanagement im Krankenhaus durch Process Owner Communities. In S. Bohnet-Joschko (Hrsg.): Wissensmanagement im Krankenhaus, Effizienz- und Qualitätssteigerungen durch versorgungsorientierte Organisation von Wissen und Prozessen. Gabler Edition Wissenschaft, 1:31-48, 2007
Bryan et al. 2002	S. Bryan, S. Holmes, D. Prostlethwaite, N. Carty: The role of integrated care pathways in improving the client experience, Professional Nurse, 2002, 18(2): 77-79
Campbell et al. 1998	H. Campbell, R. Hotchkiss, N.Bradshaw, M. Porteous: Integrated care pathways, British Medical Journal, 1998, 316(7125): 133-137
Casati et al. 1999	F. Casati, S. Ceri, S. Paraboschi, G. Pozzi: Specification and Implementation of Exceptions in Workflow Management Systems, ACM Trans. Database Syst. 24(3): 405-451, 1999
Casati et al. 2000	F. Casati, S. Castano, M. Fugini, I. Mirbel, B. Pernici: Using Patterns to Design Rules in Workflows. IEEE Transactions on Software Engineering, 26(8): 760-785, 2000
Cechticky et al. 2004	V. Cechticky, A. Pasetti, O. Rohlik, W. Schaufelberger: XML-based Feature-Modelling. In J. Bosch, C. Krueger (Hrsg.): ICSR 2004, Vol. 3107 of Lecture Notes in Computer Science, Springer Verlag, Berlin Heidelberg, 101-114, 2004
Chassin & Galvin 1998	M.R. Chassin, R.W. Galvin: The urgent need to improve health care quality, Institute of Medicine National Roundtable on Health Care Quality, The Journal of the American Medical Assosiation, 1998, 280: 1000-1005
Cheah 1998	J. Cheah: Clinical pathways: changing the face of client care delivery in the next millennium, Clinician in Management, 1998, 7(78): 78-84
Chiu et al. 1999	D.K.W. Chiu, Q. Li, K. Karlapalem: A Meta Modeling Ap-

	proach to Workflow Management System Supporting Exception Handling. Information Systems 24(2): 159-184, 1999
Claasbrummel et al. 2007	B. Claasbrummel, W. Deiters, K. Grunau, O. Koch, S. Meister: Abschlussbericht der Arbeitsgruppe Telemedizin - Analyse und Identifikation informationstechnischer Anwendungen und Einbindung in integrierte Versorgungsprozesse, Interner Bericht, Fraunhofer ISST, 2007
Clarke Jr. et al. 1999	E.M. Clarke Jr., O. Grumberg, D.A. Peled: Model Checking. The MIT Press, Cambridge, Massachusetts, and London, UK, 1999
Currie & Harvey 2000	V. L. Currie, G. Harvey: The use of care pathways as tools to support the implementation of evidence-based practice, Journal of Interprofessional Care, 2000, 14(4): 311-324
Czarnecki & Eisenecker 2000	K. Czarnecki, U. Eisenecker: Generative Programming – Methods, Tools, and Applications. Addison-Wesley, Boston, MA, 2000
Czarnecki & Kim 2005	K. Czarnecki, C. Kim: Cardinality-Based Feature Modeling and Constraints: A Progress Report. In Proceedings of Int'l OOPSLA Workshop on Software Factories, 26-20, 2005
Czarnecki & Pietroszek 2006	K. Czarnecki, K. Pietroszek: Verifying Feature-Based Model Templates Against Well-Formedness OCL Constraints. In Proceedings of the 5th Int'l Conf. on Generative Programming and Component Engineering (GPCE 2006), Portland, Oregon, USA, 211-220, 2006
Czarnecki et al. 2002	K. Czarnecki, T. Bednasch, P. Unger, U. Eisenecker: Generative Programming for Embedded Software: An Industrial Experience Report. In D. Batory, C. Consel, W. Taha (Hrsg.): GPCE 2002, Vol. 2487 of Lecture Notes in Computer Science, Springer Verlag, Berlin Heidelberg, 156-172, 2002
Czarnecki et al. 2004	K. Czarnecki, S. Helsen, U. W. Eisenecker: Staged configuration using feature models. In R. L. Nord (Hrsg.): Software Product Lines: Proceedings of the Third Int'l Conference (SPLC 2004), Vol. 3154 of Lecture Notes in Computer Science, Springer Verlag, Heidelberg, 266-283, 2004
Czarnecki et al. 2005	K. Czarnecki, S. Helsen, U. W. Eisenecker: Formalizing cardinality-based feature models and their specialization. Software Process:Improvement and Practice, 10(1): 7-29, 2005
Dadam et al. 2000	P. Dadam, M. Reichert, K. Kuhn: Clinical Workflows - The Killer Application for Process-oriented Information Systems?

	In: W. Abramowicz, M.E. Orlowska (Hrsg.): BIS 2000 - Proc. of the 4th Int'l Conf. on Business Information Systems, Poznan, Polen, 2000, Springer-Verlag, 2000, 36-59
Davenport 1993	T.H. Davenport: Process Innovation: Reengineering Work through Information Technology, Boston, 1993
DDCZ 2007	Das Deutsche Cochrane Zentrum: Leitlinien. http://www.cochrane.de/de/guidelines.htm, abgerufen am 15. Juni 2009
De Luc 2000	K. de Luc: Care pathways: an evaluation of their effectiveness, Journal of Advances Nursing, 2000, 32(2): 485-496
Deiters 1992	W. Deiters: A View Based Approach to Software Process Management. Dissertation, Techn. Univ. Berlin, Deutschland, 1992
Deiters et al. 1994	W. Deiters, V. Gruhn, H. Weber: Software Process Evolution in MELMAC. The Impact of CASE on the Software Development Life Cycle, Singapore u. a., 1994
Deiters et al. 1995	W. Deiters, V. Gruhn, R. Striemer: Der FUNSOFT-Ansatz zum integrierten Geschäftsprozessmanagement. Wirtschaftsinformatik (1995), Heft 5, 459-466, 1995
Deiters et al. 1996	W. Deiters, T. Löffeler, R. Striemer, T. Herrmann: Identifikation, Klassifikation und Unterstützung semistrukturierter Teilprozesse in prozessorientierten Informationssystemen, In: H. Krcmar, H. Lewe, G. Schwabe (Hrsg.): Herausforderung Telekooperation. Einsatzerfahrungen und Lösungsansätze für ökonomische und ökologische, technische und soziale Fragen unserer Gesellschaft, 1996, 261-274, Springer, Berlin
Deursen et al. 2000	A. v. Deursen, P. Klint, J. Visser: Domain-Specific Languages: An Annotated Bibliography, ACM SIGPLAN, Notices, 35(6): 26-36, 2000
DKG 2004	Deutsche Krankenhausgesellschaft (DKG): Informationen für Krankenhäuser zur integrierten Versorgung §§140a bis d SGB V, 2004
Doherty 2006	G.M. Doherty: Special Medical Problems in Surgical Patients, In: G.M. Doherty, L.W. Way (Hrsg.): Current Surgical Diagnosis & Treatment, Mcgraw-Hill Professional, 2006
Dori 2002	D. Dori: Object Process Methodology – a Holistic Systems Paradigm. Springer, Berlin, Heidelberg, New York, 2002

Drumm & Achenbach 2005	S. Drumm, A. Achenbach: Integrierte Versorgung mit Klinischen Pfaden erfolgreich gestalten. In: W. Hellmann, Ecomed Medizin, Landsberg/Lech, 2005
Eckenbach & Böckmann 2008	M. Eckenbach, B. Böckmann: DiPP - Digitale Pfade im Gesundheitsnetz Prosper - Unterstützung der Integrierten Versorgung durch IT-gestützte transsektorale Pfade, In: S. Schug, U. Engelmann (Hrsg.): Telemed 2008 Proceedings, Berlin, 2008
Ehrig et al. 2006	H. Ehrig, K. Ehrig, U. Prange, G. Taentzer: Fundamentals of Algebraic Graph Transformation. EATCS Monographs in Theoretical Computer Science, Springer 2006
European Pathway Association 2005	European Pathway Association: Clinical / Care Pathways. Slovenian Board Meeting, 2005, http://www.e-p-a.org/000000979b08f9803/index.html, abgerufen am 15. Juni 2009
Fahland 2007	D. Fahland: Towards Analyzing Declarative Workflows. In J. Köhler, M. Pistore, A.P. Sheth, P. Traverso, M. Wirsing (Hrsg.): Autonomous and Adaptive Web Services, Dagstuhl Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Deutschland, 2007
Fardon 2001	D.F. Fardon: Nomenclature and classification of lumbar disc pathology: Recommendations of the combined task forces of the North American Spine Society, American Society of Spine Radiology & American Society of Neuroradiology, Spine 2001, 26: 2521-2532
Fettke & Loos 2003	P. Fettke, P. Loos: Classification of Reference Models - a Methodology and its Appliaction, Information Systems and e-Business Management, 1(1):35-53, 2003
Fettke et al. 2006	P. Fettke, P. Loos, J. Zwicker: Business Process Reference Models: Survey and Classification, In C. Bussler, A. Haller (Ed.): Business Process Management Workshops, vol. 3812 of Lecture Notes in Computer Science, 469-483, Berlin Heidelberg, 2006, Springer Verlag
Fischer 1997	W. Fischer: Patientenklassifikationssysteme zur Bildung von Behandlungsfallgruppen im stationären Bereich. Prinzipien und Beispiele, Bundesamt f. Sozialversicherung, Bern (Hrsg.), ZIM-Verlag Zentrum für Informatik und wirtschaftliche Medizin, 1997, S. 211 ff und S. 473 ff
Fischer 2001	W. Fischer: Grundzüge von DRG-Systemen, In: Kranken-

	hausreport 2000, Schattauer Verlagsgesellschaft, Stuttgart, 2001
Fox et al. 2003	R. Fox, S. Moran, A. MacCormick: Guidance for integrated care pathways: a reference document for an acute NHS trust, Journal of Integrated Care Pathways, 2003, 7: 100-106
Gaitanides 1983	M. Gaitanides: Prozessorganisation, Entwicklung, Ansätze und Programme prozessorientierter Organisationsgestaltung, München 1983
Georgakopoulos et al. 1995	D. Georgakopoulos, M.F. Hornick, A.P. Sheth: An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure, Distributed and Parallel Databases, (3)2: 119-153, 1995
Gottschalk et al. 2008	F. Gottschalk, W.M.P. van der Aalst, M.H. Jansen-Vullers, M. La Rosa: Configurable Workflow Models. Int'l. Journal of Cooperative Information Systems, 17(2): 223-225, 2008
Greiling et al. 2003	M. Greiling, J. Mormann, R. Westerfeld: Klinische Pfade steuern. Baumann Fachverlag, Kulmbach, 2003
Greitemann & Stein 2005	B. Greitemann, V. Stein: Rehabilitation bei Bandscheibenvorfall mit radikulärer Symptomatik und nach Bandscheibenoperation, AWMF online, http://www.uni-duesseldorf.de/awmf/ll/033-048.htm, abgerufen am 10. Juli 2009
Gruhn 1991	V. Gruhn: Validation and Verification of Software Process Models. Dissertation, Univ. Dortmund, Deutschland, 1991
Hagemeyer et al. 1997	J. Hagemeyer, T. Herrmann, K. Just, R. Striemer: Flexibilität bei Workflow-Management-Systemen. Tagungsband zur GI Tagung Software-Ergonomie, 179-190, Teubner, Stuttgart, 1997
Hallerbach 2009	A. Hallerbach: Management von Prozessvarianten. Dissertation, Universität Ulm, 2009
Hallerbach et al. 2008a	A. Hallerbach, T. Bauer, M. Reichert: Context-based Configuration of Process Variants, 3rd Int'l. Workshop on Technologies for Context-Aware Business Process Management (TCoB 2008), Spain, 2008
Hallerbach et al. 2008b	A. Hallerbach, T. Bauer, M. Reichert: Anforderungen an die Modellierung und Ausführung von Prozessvarianten, Datenbank Spektrum, 24: 48-58, 2008

Hallerbach et al. 2009a	A. Hallerbach, T. Bauer, M. Reichert: Issues in Modeling Process Variants with Provop. In D. Ardagna, M. Mecella, J. Yang (Hrsg.): Business Process Management Workshops, vol. 17 of Lecture Notes in Business Information Processing, 56-67, Springer Berlin / Heidelberg, 2009
Hallerbach et al. 2009b	A. Hallerbach, T. Bauer, M. Reichert: Guaranteeing Soundness of Configurable Process Variants in Provop, 11th IEEE Conference on Commerce and Enterprise Computing (CEC'09), Austria, 2009
Hallerbach et al. 2010	A. Hallerbach, T. Bauer, M. Reichert: Configuration and Management of Process Variants. In J. vom Brocke, M. Rosemann (Hrsg.): Handbook on Business Process Management I, International Handbooks Information System, 2010, Part II, Springer, Berlin Heidelberg, 237-255
Halmans & Pohl 2003	G. Halmans, K. Pohl: Communicating the Variability of a Software-Product Family to Customers. Software and System Modeling, 2(1): 15-36, 2003
Hammer & Champy 1993	M. Hammer, J. Champy: Reengineering the Cooperation, New York, 1993 (Deutsche Übersetzung: Business Reengineering: Die Radikalkur für das Unternehmen. Frankfurt am Main 1996)
Han et al. 1996	J. Han, J. Himmighöfer, T. Schaaf, D. Wikarski: Management of Workflow Resources to Support Runtime Adaptability and System Evolution. Proc. 1st Int'l Conf. on Practical Aspects of Knowledge Management (PAKM'96), 218-224, Basel, Switzerland, 1996
Härder 1997	T. Härder (Hrsg.): Themenheft Workflow-Management, Informatik Forsch. Entw., Band 12, Heft 2, 1997
Heckel et al. 2002	R. Heckel, J. M. Küster, G. Taentzer: Confluence of Typed Attributed Graph Transformation Systems. In: 1st Int'l Conf. on Graph Transformation (ICGT'02), Vol. 2505 of Lecture Notes in Computer Science, Springer Verlag, Berlin Heidelberg, 161-176, 2002
Heinl et al. 1999	P. Heinl, S. Horn, S. Jablonski, J. Neeb, K. Stein, M. Teschke: A Comprehensive Approach to Flexibility in Workflow Management Systems. Proc. Of the Int'l joint Conference on Work Activities Coordination and Collaboration (WACC 1999), 78-88, San Francisco, California, USA, 1999
Hellmann & Eble 2009	W. Hellmann, S. Eble (Hrsg.): Gesundheitsnetzwerke managen. Kooperationen erfolgreich steuern. Medizinisch Wissen-

	schaftliche Verlagsgesellschaft Berlin, 2009
Hellmann 2002	W. Hellmann (Hrsg.): Klinische Pfade. Konzepte, Umsetzung, Erfahrungen. Ecomed Verlagsgesellschaft, Landsberg/Lech, 2002
Hill 2001	M. Hill: The Development of Care Management: Systems To Achieve Clinical Integration, In: N. Faass (Hrsg.): Integrating Complementary Medicine into Health Systems, Jones and Bartlett Publishers Inc., 2001, 249-256
Hindle & Yazbeck 2005	D. Hindle, A.-M. Yazbeck: Clinical pathways in 17 European Union countries: a purposive survey, 2005, Australian Health Review, 29(1): 94-104
Hindle 2007	D. Hindle: Nutzen, Grenzen und Potentiale Klinischer Behandlungspfade – ausgewählte Studienergebnisse. In N. Roeder, T. Küttner (Hrsg.): Klinische Behandlungspfade, Mit Standards erfolgreicher arbeiten. Deutscher Ärzteverlag, 71-78, 2007
Hoffman 1993	P. A. Hoffman: Critical Path Method: An Important Tool for Coordinating Clinical Care, Joint Commission Journal on Quality Improvement, 1993, 19(7): 235-246
Holler et al. 2002	T. Holler, K. Bissat, H. Müller, C. Reemts, E. Rieben, K. Schmid: Praktische Pfadarbeit - Konstruktion, Implementierung und Controlling von Patientenpfaden, In W. Hellmann (Hrsg.): Klinische Pfade. Konzepte, Umsetzung, Erfahrungen, 2002, Ecomed, Landsberg/Lech
Hollingsworth 1995	D. Hollingsworth: The Workflow Reference Model, WfMC TC00-1003, Version 1.1, Technischer Bericht, Workflow Management Coalition, 1995
Holzmann 2003	G.J. Holzmann: The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley, Boston, Massachusetts, USA, 2003
Horn & Jablonski 1998	S. Horn, S. Jablonski: An Approach to Dynamic Instance Adaption in Workflow Management Applications. Proc. Workshop towards Adaptive Workflow Systems of ACM 1998 Conference on Computer Supported Cooperative Work (CSCW'98), Seattle, WA, USA, 1998
Hsu 1993	M. Hsu (Hrsg.): Special Issue on Workflow and Extended Transaction Systems, IEEE Bulletin of the Technical Commitee on Data Engineering, Vol. 16, No. 2, 1993

Hsu 1995	M. Hsu (Hrsg.): Special Issue on Workflow Systems, IEEE Bulletin of the Technical Commitee on Data Engineering, Vol. 18, No. 1, 1995
Hull et al. 1997	R.D. Hull, G.E. Raskob, R.F. Brant, G.F. Pineo, K.A. Valentine: The Importance of Initial Heparin Treatment on Long-term Clinical Outcomes of Antithrombotic Therapy. The Emerging Theme of delayed Reccurence, Arch. Intern. Med. 1997, 157(20): 2317-2321
ISO 2005	ISO: Unified Modeling Language Specification, Version 1.4.2, formal/05-04-01, ISO/IEC 19501:2005(E), 2005, http://www.omg.org/cgi-bin/doc?formal/05-04-01.pdf, abgerufen am 19. Juli 2010
Jablonski & Bussler 1996	S. Jablonski, C. Bussler: Workflow Management - Modeling Concepts, Architecture, and Implementation, International Thompson Computer Press, 1996
Jablonski et al. 1997a	S. Jablonski, K. Stein, M. Teschke: Experiences in Workflow Management for Scientific Computing. Proc. 8th Int'l Workshop on Database and Expert Systems Applications (DEXA'97), 56-61, Toulouse, France, 1997
Jablonski et al. 1997b	S. Jablonski, M. Böhm, W. Schulze (Hrsg.): Workflow-Management: Entwicklung von Anwendungen und Systemen, Heidelberg, Dpunkt-Verlag, 1997
Joeris 1999	G. Joeris: Defining Flexible Workflow Execution Behaviors. Proc. Workshop on Enterprise-wide and Cross-Enterprise Workflow-Mgmt., 49-55, Deutschland, 1999
Jones 2000	A. Jones: Implementation of hospital care pathways for clients with schizophrenia. Journal of Nursing Management 8(40): S. 215-225, 2000
Kang et al. 1990	K. Kang, S. Cohen, J. Hess, Nowak, W., S. Peterson: Feature-oriented domain analysis (FODA) feasibility study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA, 1990
Kang et al. 1998	K. Kang, S. Kim, J. Lee, K. Kim: FORM: A Feature-Oriented Reuse Method. In Annals of Software Engineering 5: 143.168, 1998
Kearon 2003	C. Kearon: Natural history of venous thromboembolism. Circulation 2003, 107: I-22-I30

Keun & Prott 2008	F. Keun, R. Prott: Einführung in die Krankenhaus-Kostenrechnung. Anpassung an neue Rahmenbedingungen, Gabler Verlag 2008, S. 223-224
Kifer et al. 1995	M. Kifer, G. Lausen, J. Wu: Logical foundations of object-oriented and frame-based languages. Journal of the ACM, 42: 741-843, 1995
Kleemann 2010	T. Kleemann: Die dritte Generation von Krankenhausinformationssystemen – Workflowunterstützung und Prozessmanagement. In H. Schlegel (Hrsg.): Steuerung der IT im Klinikmanagement – Methoden und Verfahren. 1. Auflage 2010, Vieweg+Teubner Verlag, 267-276, 2010
Krämer 2006	J. Krämer: Bandscheibenbedingte Erkrankungen, Thieme, Stuttgart, 2006
Kreuzer 2006	R. Kreuzer: Komplikationen bei posteriorer lumbaler intercorporeller Fusion, Dissertation, Universitätsmedizin Berlin, 2006
Kurt 2006	A. Kurt: Die klinischen Langzeitfolgen der intraoperativen Duraverletzung in der Wirbelsäulenchirurgie, Dissertation, Ruhr-Univ. Bochum, 2006
La Rosa 2009	M. La Rosa: Managing Variability in Process-Aware Information Systems, Disseration, Queensland Univ. of Technology, Brisbane, Australia
Lenz & Kuhn 2004	R. Lenz, K. Kuhn: Aspekte einer prozessorientierten Systemarchitektur für Informationssysteme im Gesundheitswesen, GI Jahrestagung (2), 2004: 530-536
Lenz & Reichert 2007	R. Lenz, M. Reichert: IT support for healthcare processes - premises, challenges, perspectives, Data & Knowledge Engineering, Vol. 61, No. 1, 2007, 39-58
Liu & Pu 1997	L. Liu, C. Pu: ActivityFlow: Towards Incremental Specification and Flexible Coordination of Workflow Activities. Proc. 16th Int'l Conf. on Conceptual Modeling (ER'97), Lecture Notes in Computer Science, vol. 1331, 169-182, Springer Berlin / Heidelberg, 1997
Lowe 1998	C. Lowe: Care pathways: have they a place in 'the new National Health Service'?, Journal of Nursing Management, 1998, 6(5): 303-306
Lu & Sadiq 2006	R. Lu, S. Sadiq: On Managing Process Variants as an Information Resource. Technischer Bericht No. 464, School of

	Information Technology & Electrical Engineering and University of Queensland, Brisbande, Juni, 2006
Luttman 2000	R. Luttman: Variance Management Systems, Online Artikel, 2000, http://www.robertluttman.com/variance_short_take.html, abgerufen am 15. Juli 2009
Luttman et al. 1995	H.R. Luttman, G. Laffel, S. Pearson: Using PERT/CPM to design and improve clinical processes, Quality Manage Health Care, 1995, 3: 1-13
Ly et al. 2010	L. T. Ly, S. Rinderle-Ma, K. Göser, P. Dadam: On Enabling Integrated Process Compliance with Semantic Constraints in Process Management Systems. Information System Frontiers, 2010
Mangan & Sadiq 2002	P. Mangan, S. Sadiq: On Building Workflow Models for Flexible Processes. Australian Computer Science Communications, 24(2): 103-109, IEEE Computer Society Press, Los Alamitos, CA, USA, 2002
Mangan & Sadiq 2003	P. Mangan, S. Sadiq: A Constraint Specification Approach to Building Flexible Workflows. Journal of Research and Practice in Information Technology, 35(1): 21-39, Australian Computer Soc. Inc., 2003
Mannion 2002	M. Mannion: Using First-Order Logic for Product Line Model Validation. In G. Chastek (Hrsg.): Proc. Int'l Software Product Line Conference 2002, Vol. 2379 of Lecture Notes in Computer Science, Springer Verlag, Berlin Heidelberg, 176-187, 2002
McNeil 2001	B.J. McNeil: Shattuck Lecture - Hidden barriers to improvement in the quality of care, The New England Journal of Medicine, 2001, 345: 1612-1620
Meiler 2005	C. Meiler: Modellierung, Planung und Ausführung Klinischer Pfade. Eine integrierte modellbasierte Betrachtung evidenzbasierter, organisatorischer und betriebswirtschaftlicher Gesichtspunkte, Dissertation, Univ. Erlangen-Nürnberg, 2005, S. 24-37
Merriam-Webster 2003	Merriam-Webster's Eleventh Collegiate Dictionary, Merriam-Webster, Inc. Springfield, Massachusetts, USA, 2003
Merritt et al. 1999	T. A. Merritt, M. Gold, J. Holland: A critical evaluation of clinical practice guidelines in neonatal medicine: does their use improve quality and lower costs?, Journal of Evaluation in

	Clinical Practice, 1999, 5(2): 169-177
Mikulaninec 1992	C. E. Mikulaninec: An Amputee Critical Path, Journal of Vascular Nursing, 1992, 10(2): 6-9
Minor et al. 2007a	M. Minor, A. Tartakovski, R. Bergmann: Representation and Structure-based Similarity Assessment for Agile Workflows. In Proceedings of the 7th Int'l Conf. on Case-Based Reasoning: Case-Based Reasoning Research and Development (ICCBR'07). Lecture Notes in Artificial Intelligence, vol. 4626, 224-238, Springer-Verlag, Berlin / Heidelberg, 2007
Minor et al. 2007b	M. Minor, D. Schmalen, A. Koldehoff, R. Bergmann: Structural Adaptation of Workflows Supported by a Suspension Mechanism and by Case-based Reasoning. In: Proceedings of the 16th IEEE Int'l Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'07), 370-375, 2007
Müller & Heller 1998	R. Müller, B. Heller: A petri net-based model for knowledge-based workflows in distributed cancer therapy. Proc. Of the EDBT'98 Workshp on Workflow Management Systems, 91-99, 1998
Müller & Rahm 1999	R. Müller, E. Rahm: Rule-Based Dynamic Modification of Workflows in a Medical Domain. Proc. Datenbanksysteme in Büro, Technik und Wissenschaft (BTW'99), 429-448, Freiburg, Deutschland, 1999
Müller et al. 2001	H.P. Müller, K. Schmid, D. Conen: Interne Leitlinien und Patientenpfade, In: Medizinische Klinik, Ausgabe (96)11, 2001, S. 692-697
Müller et al. 2004	R. Müller, U. Greiner, E. Rahm: AGENTWORK: a workflow system supporting rule-based workflow adaptation, Data & Knowledge Engineering, 51(2): 223-256, 2004
Mulyar et al. 2008	N. Mulyar, M. Pesic, W.M.P. van der Aalst, M. Peleg: Declarative and Procedural Approaches for Modelling Clinical Guidelines: Adressing Flexibility Issues. In M. Reichert, R. Lenz, and M. Peleg (Hrsg.): Informal Proceedings of the Int'l Workshop on Process-Oriented Information Systems in Healthcare (ProHealth 2007), 17-18, Australia 2007
Neuhaus et al. 2010	J. Neuhaus, S. Houta, C. Reuter: Neue Ansätze bei der Umsetzung von Behandlungspfaden - Flexibilisierungskonzepte am Beispiel der Behandlung von Wirbelsäulenerkrankungen. In W. Hellmann, S. Eble (Hrsg.): Ambulante und Sektoren übergreifende Behandlungspfade, Wissenschaftlich medizinische Verlagsgesellschaft, 79-97, 2010

Nordsieck 1934	F. Nordsieck: Grundlagen der Organisationslehre, Stuttgart, 1934
Nüllen & Noppeney 2006	H. Nüllen, T. Noppeney: Lehrbuch Qualitätsmanagement in der Arztpraxis: Entwicklung und Einführung eines QMS, Deutscher Ärzte-Verlag, 2006
OASIS 2007	OASIS: Web Services Business Process Execution Language Version 2.0. OASIS Standard, April 2007, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html, abgerufen am 12. Mai 2010
OMG 2009	OMG: Business Process Model and Notation (BPMN) FTF Beta 1 Version 2.0. August 2009, http://www.omg.org/spec/BPMN/2.0/Beta1/PDF, abgerufen am 12. Mai 2010
OMG 2010	OMG: Object Constraint Language Version 2.2. http://www.omg.org/spec/OCL/2.2/PDF, abgerufen am 15.02.2010
Paeger 2001	A. Paeger, Aufsichtsratsmitglied der Ameos AG und Vorstand der Ameos Holding AG, ehem. Hauptgeschäftsführer Asklepios zitiert nach Stockhorst: http://www.klinikum-hannover.de/arztd/veran/clipath.pdf, abgerufen am 15. Juni 2009
Panella et al. 2003	M. Panella, S. Marchisio, F. Di Stanislao: Reducing clinical variations with clinical pathways: do pathways work?, International Journal for Quality in Health Care, 2003, 15(6): 509-521
Pesic & van der Aalst 2006	M. Pesic, W.M.P. van der Aalst: A Declarative Approach for Flexible Business Process Management. In J. Eder, S. Dustdar (Hrsg.): BPM 2006 Workshops, Workshop on Dynamic Process Management (DPM 2006), vol. 4103 of Lecture Notes in Computer Science, 81-92, Springer-Verlag, Berlin / Heidelberg, 2006
Pesic et al. 2007	M. Pesic, M.H. Schonenberg, N. Sidorova, W.M.P. van der Aalst: Constraint-Based Workflow Models: Change Made Easy. In F. Curbera, F. Leymann, M. Weske (Hrsg.): Proc. of the OTM Conference on Cooperative Information Systems (CooPIS 2007), vol. 4803 of Lecture Notes in Computer Science, 77-94, Springer-Verlag, Berlin / Heidelberg, 2007
Polyvyanyy & Weske 2009	A. Polyvyanyy, M. Weske: Hypergraph-based Modeling of Ad-Hoc Business Processes. In D. Ardagna, M Mecella, J. Yang (Hrsg.): BPM 2008 Workshops, LNBIP 17 , 278-289, Springer-Verlag Berlin Heidelberg, 2009

Porter 1989	M.E. Porter: Wettbewerbsvorteile, Franfurt am Main, 1989 (Original: Competitive Advantage. Creating and Sustaining Superior Performance, New York, 1985)
Puhlmann et al. 2005	F. Puhlmann, A. Schnieders, J. Weiland, M. Weske: Variability Mechanisms for Process Models. PESOA-Report TR 17/2005, Process Familiy Engineering in Service-Oriented Applications (PESOA), 2005
Reichert & Dadam 1998	M. Reichert, P. Dadam: $ADEPT_{flex}$ - Supporting Dynamic Changes of Workflows Without Losing Control, Journal of Intelligent Information Systems 10, 93-128, 1998
Reichert & Dadam 2000	M. Reichert, P. Dadam: Geschäftsprozessmodellierung und Workflow-Management - Konzepte, Systeme und deren Anwendung, Industrie Management, (Themenheft: Modellierung und Simulation), 16(3): 23-27, 2000
Reichert 2000a	M. Reichert: Prozessmanagement im Krankenhaus - Nutzen, Anforderungen und Visionen, das Krankenhaus, 92(11): 903-909, 2000
Reichert 2000b	M. Reichert: Dynamische Ablaufänderungen in Workflow-Management-Systemen, Dissertation, Univeristät Ulm, 2000
Reichert et al. 1997	M. Reichert, B. Schultheiß, P. Dadam: Erfahrungen bei der Entwicklung vorgangsorientierter, klinischer Anwendungssysteme auf Basis prozessorientierter Workflow-Technologie, Proc. 42. Jahrestagung der GMDS, Ulm, Deutschland, 1997
Reichert et al. 2002	M. Reichert, T. Bauer, T. Fries, P. Dadam: Modellierung planbarer Abweichungen in Workflow-Management-Systemen. In M. Glinz, G. Müller-Luschnat (Hrsg.): Proc. Modellierung 2002, 183-194, Arbeitstagung der GI, Tutzing, Deutschland (GI-Edition Lecture Notes in Informatics, Band P-12), 2002
Reichert et al. 2005	M. Reichert, S. Rinderle, U. Kreher, P. Dadam: Adaptive Process Management with ADEPT2, Proc. of the Int'l. Conf. on Data Engineering (ICDE'05), 1113-1114, Tokyo, Japan
Reuter & Neuhaus 2008	C. Reuter, J. Neuhaus: Referenzprozesse und deren Use Cases - Spezifikation einer Architektur zum sicheren Austausch von Patientendaten, Spezifikation der elektronischen FallAkte (eFA), Fraunhofer ISST, 2008
Reuter 2009	C. Reuter: SPOT Demonstrator – Referenzprozesse, Use Cases und konzeptionelle Lösung. Fraunhofer ISST, interner

	Bericht, 2009
Reuter 2010	C. Reuter: Composition of Semantic Process Fragments to Domain-related Process Families. In P. van Bommel, S. Hoppenbrouwers, S. Overbeek, E. Proper, J. Barjis (Hrsg.): The Practice of Enterprise Modeling (PoEM 2010), LNBIP 68, S. 61-75, Springer Berlin-Heidelberg, 2010
Reuter et al. 2009	C. Reuter, T. Königsmann, S. Meister, S. Houta, J. Neuhaus: CHOPIN: Toolbox for Composition of Telemedical Services. In Proceedings of the 2nd IASTED Int'l Conf on Telehealth and Assistive Technology, TAT 2009, ACTA Press, 2009
Reuter et al. 2010	C. Reuter, S. Houta, J. Neuhaus: Prozessorientierte Bereitstellung telemedizinischer Anwendungen an der Schnittstelle Arzt-Patient. In F. Duesberg (Hrsg.): e-Health 2011, Informationstechnologien und Telematik im Gesundheitswesen, medical future verlag, 150-155, 2010
Riebisch 2003	M. Riebisch: Towards a More Precise Definition of Feature Models. In M. Riebisch, J.O. Cplien, D. Streitferdt (Hrsg.): Modeling Variability for Object-Oriented Product Lines, 2003
Riebisch et al. 2002	M. Riebisch, K. Böllert, D. Streitferdt, I. Philippow: Extending Feature Diagrams with UML Multiplicities. In Proceedings of the Sixth Conf. on Integrated Design and Process Technology (IDPT 2002), Pasadena, Kalifornien, USA, 2002
Riebisch et al. 2004	M. Riebisch, D. Streitferdt, I. Pashov: Modeling variability for object-oriented product lines – workshop report. In F. Buschmann, A.P. Buchmann, M. A. Cilia (Hrsg.): Object-Oriented Technology, ECOOP 2003 Workshop Reader. Vol. 3013 of Lecture Notes in Computer Science, Springer Verlag, Heidelberg, 165-178, 2004
Riehle & Züllighoven 1996	D. Riehle, H. Züllighoven: Understanding and Using Patterns in Software Development. Theory and Practice of Object Systems, 2(1): 3-13, 1996
Rinderle 2004	S. Rinderle: Schema Evolution in Process Management Systems. Dissertation, Universität Ulm, 2004
Rinderle et al. 2002	S. Rinderle, M. Reichert, P. Dadam: Effiziente Verträglichkeitsprüfung und automatische Migration von Workflow-Instanzen bei der Evolution von Workflow Schemata, Informatik Forschung Entwicklung, 17: 177-197, 2002
Rinderle et al. 2003	S. Rinderle, M. Reichert, P. Dadam: Evaluation of Correctness Criteria for Dynamic Workflow Changes, In: Proc. 1st

	International Conference on Business Process Management (BPM '03), Eindhoven, Niederlande
Rinderle et al. 2004	S. Rinderle, M. Reichert, P. Dadam: Flexible support of team processes by adaptive workflow systems. Distrib. Parallel Databases 16: 91-116, 2004
Robinson et al. 1992	J. A. Robinson, K. J. Robinson, D. J. Lewis: Balancing Quality of Care and Cost-Effectiveness through Case Management, ANNA Journal, 1992, 19(2): 182-188
Roeder et al. 2003a	N. Röder, P. Hensen, D. Hindle, N. Loskamp, H.-J. Lakornek: Instrumente zur Behandlungsoptimierung - Klinische Behandlungspfade. In: Der Chirurg. Ausgabe (74)12: 1149-1155, 2003
Roeder et al. 2003b	N. Roeder, D. Hindle, N. Loskamp, C. Juhra, P. Hensen, H. Bunzemeier, B. Rochell: Frischer Wind mit klinischen Behandlungspfaden (I). Instrumente zur Verbesserung der Organisation klinischer Prozesse, das Krankenhaus, 2003, 1:20-27
Rosemann & van der Aalst 2007	M. Rosemann, W.M.P. van der Aalst: A Configurable Reference Modelling Language. Information Systems, 32: 1-12, 2007
Rudolph 2010	S. Rudolph: Semantische Komposition von Prozessfragmenten im Umfeld dynamischer Prozessmodellierung. Diplomarbeit, Technische Universität Dortmund, 2010
Russell et al. 2004a	N. Russell, A.H.M. ter Hofstede, D. Edmond, W.M.P. van der Aalst: Workflow Data Patterns, QUT Technical report, FIT-TR-2004-01, Queensland University of Technology, Brisbane, Australien, 2004
Russell et al. 2004b	N. Russell, A.H.M. ter Hofstede, D. Edmond, W.M.P. van der Aalst: Workflow Resource Patterns, BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven, Niederlande, 2004
Russell et al. 2006a	N. Russell, A.H.M. ter Hofstede, W.M.P. van der Aalst, N. Mulyar: Workflow Control-Flow Patterns: A Revised View, BPM Center Report BPM-06-22, BPMcenter.org, 2006
Russell et al. 2006b	N. Russell, W.M.P. van der Aalst, A.H.M. ter Hofstede: Exception Handling Patterns in Process-Aware Information Systems, BPM Center Report BPM-06-04, BPMcenter.org, 2006
Sachverständigenrat	Sachverständigenrat für die Konzertierte Aktion im Gesund-

2010	heitswesen: Bedarfsgerechtigkeit und Wirtschaftlichkeit, Band III, Über- Unter- und Fehlversorgung. Gutachten 2000/2001, Ausführliche Zusammenfassung, http://www.svr-gesundheit.de/Gutachten/Gutacht01/Kurzf-de.pdf, abgerufen am 29. Juli 2010
Sadiq et al. 2001	S. Sadiq, W. Sadiq, M. Orlowska: Pockets of Flexibility in Workflow Specification. Proc. Of the 20th Int'l Conference on Conceptional Modeling, Lecture Notes in Computer Science, vol. 2224, 513-526, Springer Berlin / Heidelberg, 2001
Sadiq et al. 2005	S. Sadiq, M. Orlowska, W. Sadiq: Specification and Validation of Process Constraints for Flexible Workflows. In: Information Systems 30 (2005), 349-378, 2005
Salfeld et al. 2009	R. Salfeld, S. Hehner, R. Wichels: Modernes Krankenhaus-management – Konzepte und Lösungen. 2. Auflage, Sprin-ger-Verlag Berlin Heidelberg, 2009
Sanjiva et al. 2005	W. Sanjiva, F. Curbera, F. Leymann, T. Storey, D.F. Fergu-son: Web Service Platform Architecture, Pearson Education, 2005
Scheer 1998	A.-W. Scheer: »ARIS - House of Business Engineering«: Konzept zur Beschreibung und Ausführung von Referenzmo-dellen, In J. Becker, M. Rosemann, R. Schütte: Referenzmo-dellierung, 2-21, Physica-Verlag Heidelberg, 1998
Scheffer 1996	T. Scheffer: Algebraic foundation and improved methods of induction of ripple down rules. In Proceedings Pacific Rim Workshop on Knowledge Acquisition, Sydney, Australia, 1996
Schilling et al. 2006	M. K. Schilling, S. Richter, P. Jacob, W. Lindemann: Klinische Behandlungspfade, Erste Ergebnisse des systematischen IT-gestützten Einsatzes an einer chirurgischen Universitätsklinik. Dtsch Med Wochenschr 2006, Georg Thieme Verlag KG Stuttgart, 131: 962-967
Schobbens et al. 2006	P.-Y. Schobbens, P. Heymans, J.-C. Trigaux: Feature Dia-grams: A Survey and a Formal Semantics. Proc. 14th IEEE Int'l Requirements Engineering Conf. (RE'06), Washington, DC (USA), 136-145, 2006
Schonenberg et al. 2008	H. Schonenberg, R. Mans, N. Russell, N. Mulyar, W.M.P. van der Aalst: Process Flexibility: A Survey of Contemporary Ap-proaches. In J.L.G. Dietz, A. Albani, J. Barjis (Hrsg.): Ad-vances in Enterprise Engineering I, vol. 10 of Lecture Notes in Business Information Processing, 16-30, Springer Berlin / Heidelberg, 2008

Schütte 1997	R. Schütte: Foundations on Reference Modeling, PhD Thesis, 1997, Univ. Münster
Seelig & Nidecker 1989	W. Seelig, A. Nidecker: Schmerzen nach Operationen an der Lendenwirbelsäule. Das »Failed Back Syndrom«, Zeitschrift für Orthopädie 127: 346-353, 1989
Seifried & Heinrich 2000	E. Seifried, F. Heinrich: Lungenembolie, Thieme-Verlag, 2000
Simpson et al. 1993	J.M. Simpson, C.P. Silveri, R.A. Balderston, F.A. Simeone, H.S. An: The results of operations on the lumbar spine in patients who have diabetes mellitus, Journal of Bone Joint Surgery, 75:1823-1829, 1993
Soffer et al. 2003	P. Soffer, B. Golany, D. Dori: ERP modeling: a comprehensive approach. In Information Systems 28: 673-690, 2003
Streitferdt et al. 2003	D. Streitferdt, M. Riebisch, I. Philippow: Details of formalized relations in feature models using OCL. In Proceedings of the 10th IEEE Int'l Conf. on Engineering of Computer-Based Systems (ECBS 2003), 297-304, 2003
Strong & Miller 1995	D.M. Strong, S.M. Miller: Exceptions and Exception Handling in Computerized Information Processes. ACM Transactions on Information Systems, 13(2): 206-233, 1995
Sun et al. 2005	J. Sun, H. Zhang, Y. F. Li, H. Wang: Formal Semantics and Verification of Feature Modeling. Proceedings of the 10th IEEE Int'l Conf. on Engineering of Complex Systems (ICECCS'05), Washington, DC (USA), 303-312, 2005
Svahnberg et al. 2005	M. Svahnberg, J. van Gurp, J. Bosch: A taxonomy of variability realization techniques. Software - Practice and Experience, 35:705-754, 2005
Tenckhoff 2003	B. Tenckhoff: Modulare Clinical Pathways – Konzept und Internetforum ClinPath.de. In: W. Hellmann (Hrsg.): Praxis Klinischer Pfade. Viele Wege führen zum Ziel. ecomed Landsberg Heidelberg, 192-213, 2003
ter Hofstede et al. 2010	A.H.M. ter Hofstede, W.M.P. van der Aalst, M. Adams, N. Russell (Hrsg.): Modern Business Process Automation. YAWL and its Support Environment. Springer-Verlag Berlin Heidelberg, 2010
Thiel et al. 2006	R. Thiel, K. Eisenblätter, M. Kurzidem, P. Hutmacher, A. Herde, H.A. Müller: Klinische Behandlungspfade, Einführung in

	einer urologischen Klinik. Urologe 2006, Springer Medizin Verlag, 45:1415-1423, 2006
Thiemann 1996	H. Thiemann: Clinical Pathways, Instrument zur Qualitätssicherung. In f&w, Ausgabe 13(5): 454, 1996
Van der Aalst & Pesic 2006	W.M.P. van der Aalst, M. Pesic: DecSerFlow: Towards a Truly Declarative Service Flow Language. In: M. Bravetti, M. Núnez, G. Zavattaro (Hrsg.): Web Services and Formal Methods (Proc. of the 3rd Int'l. Workshop, WS-FM 2006, Austria), Lecture Notes in Computer Science, Springer-Verlag, Berlin / Heidelberg, 1-23, 2006
Van der Aalst & ter Hofstede 2002	W.M.P. van der Aalst, A.H.M. ter Hofstede: YAWL: Yet Another Workflow Language. QUT Technical report, FIT-TR-2002-06, Queensland University of Technolog, Brisbane, 2002
Van der Aalst & ter Hofstede 2005	W.M.P. van der Aalst, A.H.M. ter Hofstede: YAWL: Yet Another Workflow Language (Revised Version). Information Systems, 30(4): 245-275, 2005
Van der Aalst & van Hee 2004	W.M.P. van der Aalst, K.M. van Hee: Workflow Management: Models, Methods, and Systems (Cooperative Information Systems), MIT Press, 2004
Van der Aalst 1997	W.M.P. van der Aalst: Verification of workflow nets, In: W.M.P. van der Aalst (Hrsg.): Application and Theory of Petry Nets, Bd. LN-CS 1248, 407-426, Springer-Verlag Berlin Heidelberg, 1997
Van der Aalst 1998	W.M.P. van der Aalst: The Application of Petri Nets to Workflow Management, The Journal of Circuits, Systems, and Computers, 8(1): 21-66, 1998
Van der Aalst et al. 2003	W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, A.P. Barros: Workflow Patterns, Distributed and Parallel Databases, 14(3), 5-51, 2003
Van der Aalst et al. 2004	W.M.P. van der Aalst, L. Adred, M. Dumas, A.H.M. ter Hofstede: Design and Implementation of the YAWL System. Proc. of the 16th Int. Conf. on Advanced Information Systems Engineering (CAiSE 04), Lettland, 2004
Van Deursen & Klint 2002	A. van Deursen, P. Klint: Domain-Specific Language Design Requires Feature Descriptions. Journal of Computing and Information Technology, 10(1): 1-17, 2002
Van Gurp et al.	J. van Gurp, J. Bosch, M. Svahnberg: On the Notion of Varia-

2001	bility in Software Product Lines. In Proceedings of the Working IEEE/IFIP Conf. on Software Architecture (WICSA'01), 2001
Vanhaecht et al. 2006	K. Vanhaecht, M. Bollmann, K. Bower, C. Gallagher, A. Gardini, J. Guezo, U. Jansen, R. Massoud, K. Moody, W. Sermeus, R. Van Zelm, C. Whittle, A.-M. Yazbeck, K. Zander, M. Panella: Prevalence and use of clinical pathways in 23 countries - an international survey by the European Pathway Association, 2006, Journal of Integrated Care Pathways, 10(1): 28-34
Vogel et al. 2002	S. Vogel, I. Seyfarth-Metzger, E. Höcherl: Die Entwicklung von Patientenpfaden (Clinical Pathways) im Krankenhaus München Schwabing (KMS). In: R. Burk, W. Hellmann (Hrsg.): Krankenhausmanagement für Ärztinnen und Ärzte. 1. Ergänzungslieferung, Ecomed, Landsberg, 2002
Vossen & Becker 1996	G. Vossen, J. Becker: Geschäftsprozessmodellierung und Workflo-Management, Thomson Publ., 1996
Walsh 1997	M. Walsh: Will critical pathways replace the nursing process? Nursing Standard 11(52): S. 39-42, 1997
Weber et al. 2007	B. Weber, S. Rinderle-Ma, M. Reichert: Change Support in Process-Aware Information Systems – A Pattern-Based Analysis. Technical Report TR-CTIT-07-76, Centre for Telematics and Information Technology, Universität Twente, Enschede, ISSN 1381-3625
Weber et al. 2008	B. Weber, M. Reichert, S. Rinderle-Ma: Change Patterns and Change Support Features – Enhancing Flexibility in Process-aware Information Systems, Data & Knowledge Engineering, 66: 438-466, 2008
Weber et al. 2009	B. Weber, S. Sadiq, M. Reichert: Beyond Rigidity – Dynamic Process Lifecycle Support: A Survey on Dynamic Changes in Process-aware Information Systems. In Computer Science – Research and Development, 23(2): 47-65, 2009
Wenniger 2004	S. Wenniger: Retrospektive Untersuchungen zum Risikoprofil bei Patienten mit thromboembolischen Komplikationen nach lumbaler Discusoperation, Dissertation, 2004, Neurochirurgische Klinik und Poliklinik der Tech. Univ. München
Weske 2000	M. Weske: Workflow Management Systems: Formal Foundation, Conceptional Design, Implementation Aspects. University of Münster, Dissertation, 2000

Weske 2007	M. Weske: Business Process Management: Concepts, Methods, Technology, Springer-Verlag Berlin Heidelberg, 2007
WfMC 1999	Workflow Management Coalition: Terminology & Glossary, TC 1011 Issue 3.0, 1999
White 2003	R.H. White: The epidemiology of venous thromboembolism. Circulation 2003, 107: 1-8
Wigfield & Boon 1996	A. Wigfield, E. Boon: Critical care pathway development: the way forward, British Journal of Nursing, 1996, 5(12): 732-735
Wild et al. 2004	S. Wild, G. Roglic, A. Green, R. Sicree, H. King: Global Prevalence of Diabetes: estimates of the year 2000 and projections for 2030, Diabetes Care, 27(10): 1047-1053, 2004
Wilson 1997	J. Wilson: Integrated Care Management. The Path to Success?, 1997, Butterworth-Heinemann, Oxford
Wimmer et al. 1998	C. Wimmer, H. Gluch, M. Franzreb, M. Ogon: Predisposing factors for infection in spine surgery: a survey of 850 spinal procedures. Journal of Spinal Disorders, 11:124-128, 1998
WWWC 2005	World Wide Web Consortium: XML Path Language (XPATH) 2.0. http://www.w3.org/TR/xpath20/, abgerufen am 15.02.2010
Yellowlees 2005	P. M. Yellowlees: Successfully developing a telemedicine system. Journal of Telemedicine and Telecare, 11(7): 331-335, 2005
Zander 1991	K. Zander: What's New in Managed Care and Case Management. The New Definition, 1991, 6(2): 1-2
Zander 2002	K. Zander: Integrated care pathways: eleven international trends. Journal of Integrated Care Pathways, 2002, 6: 101-107

A Anhang und Verzeichnisse

A.1 XML-Schema für SPF-Typ-Graphen

Formel 30 XML-Schema für SPF-Typ-Graphen

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<xs:complexType name="SPFTypeGraph">
		<xs:sequence>
			<xs:element ref="rootNode"/>
			<xs:element ref="constraints" minOccurs="0"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="Node" abstract="true">
		<xs:attribute name="nodeId" type="xs:string" use="required"/>
		<xs:attribute name="nodeName" type="xs:string"
use="optional"/>
		<xs:attribute name="nodeDescription" type="xs:string"
use="optional"/>
	</xs:complexType>
	<xs:complexType name="Nodes">
		<xs:sequence>
			<xs:element ref="contextNode"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="RootNode">
		<xs:complexContent>
			<xs:extension base="Node">
				<xs:choice>
					<xs:group ref="nodes"
maxOccurs="unbounded"/>
					<xs:element ref="processFragmentId"/>
				</xs:choice>
				<xs:attribute name="cardinality" use="required"
fixed="1"/>
				<xs:attribute name="operator"
type="OperatorType" use="required"/>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:complexType name="ContextNode">
		<xs:complexContent>
			<xs:extension base="Node">
				<xs:sequence>
					<xs:group ref="nodes"
maxOccurs="unbounded"/>
				</xs:sequence>
				<xs:attribute name="cardinality"
type="Cardinality" use="required"/>
				<xs:attribute name="operator"
type="OperatorType" use="required"/>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:complexType name="SPFNode">
		<xs:complexContent>
```

```
                        <xs:extension base="Node">
                                <xs:sequence>
                                        <xs:element ref="processFragmentId"/>
                                </xs:sequence>
                                <xs:attribute name="cardinality"
type="Cardinality" use="required"/>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="Constraints">
                <xs:sequence>
                        <xs:element ref="constraint" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="Constraint">
                <xs:sequence>
                        <xs:element ref="constraintId"/>
                        <xs:element ref="type"/>
                        <xs:element ref="sourceId"/>
                        <xs:element ref="targetId"/>
                </xs:sequence>
        </xs:complexType>
        <xs:simpleType name="ConstraintType">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="REQUIREMENT"/>
                        <xs:enumeration value="SEQUENCE"/>
                        <xs:enumeration value="EXCLUSION"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="OperatorType">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="AND"/>
                        <xs:enumeration value="OR"/>
                        <xs:enumeration value="XOR"/>
                        <xs:enumeration value="OPT"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="Cardinality">
                <xs:restriction base="xs:nonNegativeInteger">
                        <xs:minInclusive value="1"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:element name="rootNode" type="RootNode"/>
        <xs:element name="contextNode" type="ContextNode"/>
        <xs:element name="spfNode" type="SPFNode"/>
        <xs:group name="nodes">
                <xs:sequence>
                        <xs:choice>
                                <xs:element ref="contextNode" minOccurs="0"
maxOccurs="unbounded"/>
                                <xs:element ref="spfNode" minOccurs="0"
maxOccurs="unbounded"/>
                        </xs:choice>
                </xs:sequence>
        </xs:group>
        <xs:element name="targetId" type="xs:string"/>
        <xs:element name="sourceId" type="xs:string"/>
        <xs:element name="spfTypeGraph" type="SPFTypeGraph"/>
        <xs:element name="constraints" type="Constraints"/>
        <xs:element name="type" type="ConstraintType"/>
        <xs:element name="constraintId" type="xs:string"/>
        <xs:element name="constraint" type="Constraint"/>
        <xs:element name="processFragmentId" type="xs:string"/>
</xs:schema>
```

A.2 XML-Repräsentation des SPF-Graphen für den klinischen Pfad zur Diagnostik von Wirbelsäulenerkrankungen

Formel 31 XML-Repräsentation des SPF-Graphen für den klinischen Pfad zur Diagnostik von Wirbelsäulenerkrankungen

```
<?xml version="1.0" encoding="UTF-8"?>
<spfTypeGraph xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="R:\Dissertation\Ausfertigung\SPFTypeGraphSche
ma\SPFTypeGraph-v2.xsd">
       <rootNode nodeId="3ffe6fab-d341-4218-a6bc-2f9582d6d668"
cardinality="1" operator="OR" nodeName="ROOT">
              <contextNode nodeId="dc90a1e2-72ce-4446-b08a-e489661fbee4"
cardinality="1" operator="OR" nodeName="Diagnostik">
                     <contextNode nodeId="8c56c6b0-b193-440c-a7f1-
7103888f5169" cardinality="1" operator="AND" nodeName="Basisdiagnostik"
nodeDescription="Grundmaßnahmen der Diagnostik">
                           <spfNode nodeId="da4b78b0-4a26-4f32-a2dc-
5aee95bf0438" cardinality="1" nodeName="Anamnese" nodeDescription="Erhebung
der Symptome und der Patientenhistorie">
                                 <processFragmentId>764fcb11-6bf3-466c-
9fdb-148fca013c23</processFragmentId>
                           </spfNode>
                           <spfNode nodeId="d47835bd-0994-4df7-bd22-
9853cd1d818a" cardinality="1" nodeName="Klinische Untersuchung"
nodeDescription="Ärztliche Überprüfung der Symptome und körperlichen
Anzeichen">
                                 <processFragmentId>f9aa75fb-a416-41b0-
9acc-aa948271828e</processFragmentId>
                           </spfNode>
                     </contextNode>
                     <contextNode nodeId="579613f6-9af2-4498-84b6-
c19443731e8e" cardinality="1" operator="OR" nodeName="Erweiterte Diagnostik"
nodeDescription="Erweiterte diagnostische Maßnahmen">
                           <contextNode nodeId="4421167e-62eb-482d-8a8e-
0d8a386c5662" cardinality="1" operator="OR" nodeName="Laboruntersuchung"
nodeDescription="Labortechnische Analysen">
                                 <spfNode nodeId="a877b2e5-a57c-4c0d-
a7c8-8b2705a1abc7" cardinality="1" nodeName="Kleines Blutbild"
nodeDescription="Komponente des Standardlabors">
                                       <processFragmentId>e6fddcd5-
0df6-448c-aad1-f1fe18ac7c22</processFragmentId>
                                 </spfNode>
                                 <spfNode nodeId="3003af94-719a-4355-
b133-75f4e4f4a0d3" cardinality="2" nodeName="Blutzuckertest"
nodeDescription="Messung des Blutzuckerspiegels">
                                       <processFragmentId>5871a47e-
0f37-4b71-af4b-710dcdd6fa3d</processFragmentId>
                                 </spfNode>
                                 <spfNode nodeId="1d2b34b3-addb-4df6-
81d0-c24ffaedfd0d" cardinality="1" nodeName="Großes Blutbild"
nodeDescription="Anfertigen eines großen Blutbildes">
                                       <processFragmentId>e6fddcd5-
0df6-448c-aad1-f1fe18ac7c22</processFragmentId>
                                 </spfNode>
                           </contextNode>
                           <contextNode nodeId="69acf385-f423-4a78-81ec-
371b511326cc" cardinality="3" operator="OR" nodeName="Radiologische
Untersuchung" nodeDescription="Maßnahmen der bildgebenden, radiologischen
Diagnostik">
                                 <contextNode nodeId="662fa0f4-f24e-
4af6-b069-338de5956c2d" cardinality="1" operator="OR" nodeName="Allgemeine
```

```
Untersuchung" nodeDescription="Allgemeine radiologische Verfahren">
                                        <spfNode nodeId="52601348-0441-
4935-97e7-c05c27f9d7d7" cardinality="1" nodeName="Röntgen"
nodeDescription="Durchführung einer Röntgenuntersuchung">

       <processFragmentId>0d45c8fb-f39d-411d-8f2e-
46e32c1c71f4</processFragmentId>
                                        </spfNode>
                                        <spfNode nodeId="8b16edea-2416-
4ae2-ae71-db4c7a916214" cardinality="1" nodeName="MRT"
nodeDescription="Durchführung einer MRT-Untersuchung">

       <processFragmentId>632e796b-e616-4911-8535-
fa0fe377a171</processFragmentId>
                                        </spfNode>
                                        <spfNode nodeId="34d8bc26-2069-
40db-af5e-4c987924020a" cardinality="1" nodeName="CT"
nodeDescription="Durchführung einer CT-Untersuchung">

       <processFragmentId>a8498702-85e9-4d86-8e36-
8136c1b2e205</processFragmentId>
                                        </spfNode>
                                    </contextNode>
                                    <contextNode nodeId="9b5ce3fa-531e-
4e12-ae45-60c412240cc8" cardinality="1" operator="XOR"
nodeName="Untersuchung der Wirbelsäule" nodeDescription="Radiologische
Diagnostik der Wirbelsäule">
                                        <spfNode nodeId="119f639e-d551-
4503-92fc-f2e21958e596" cardinality="1" nodeName="Myelographie"
nodeDescription="Durchführung einer Myelographie">

       <processFragmentId>ea5eed55-71c0-451d-8855-
f1254f18a943</processFragmentId>
                                        </spfNode>
                                        <spfNode nodeId="3147a67d-5d65-
4351-822a-0fadda30d342" cardinality="1" nodeName="Discographie"
nodeDescription="Durchführung einer Discographie">

       <processFragmentId>eaca70d7-40fb-47a9-8f92-
8bf366d5da0b</processFragmentId>
                                        </spfNode>
                                    </contextNode>
                            </contextNode>
                    </contextNode>
             </contextNode>
       </rootNode>
       <constraints>
              <constraint>
                     <constraintId>ad024f87-5654-4d1c-a745-
1eec5074c652</constraintId>
                     <type>REQUIREMENT</type>
                     <sourceId>579613f6-9af2-4498-84b6-
c19443731e8e</sourceId>
                     <targetId>8c56c6b0-b193-440c-a7f1-
7103888f5169</targetId>
              </constraint>
              <constraint>
                     <constraintId>d91c014f-0f25-45ef-9788-
a90c33dba830</constraintId>
                     <type>SEQUENCE</type>
                     <sourceId>662fa0f4-f24e-4af6-b069-
338de5956c2d</sourceId>
                     <targetId>9b5ce3fa-531e-4e12-ae45-
60c412240cc8</targetId>
```

```
            </constraint>
            <constraint>
                  <constraintId>2ee75c02-47b4-4f84-8cdb-
1286bb1681e2</constraintId>
                  <type>SEQUENCE</type>
                  <sourceId>d47835bd-0994-4df7-bd22-
9853cd1d818a</sourceId>
                  <targetId>da4b78b0-4a26-4f32-a2dc-
5aee95bf0438</targetId>
            </constraint>
            <constraint>
                  <constraintId>9b25a149-062e-49b4-8187-
6f452d24b728</constraintId>
                  <type>SEQUENCE</type>
                  <sourceId>a877b2e5-a57c-4c0d-a7c8-
8b2705a1abc7</sourceId>
                  <targetId>1d2b34b3-addb-4df6-81d0-
c24ffaedfd0d</targetId>
            </constraint>
      </constraints>
</spfTypeGraph>
```

A.3 Abbildungsverzeichnis

Abbildung 1	Prozessmanagement heute	11
Abbildung 2	Prozessmanagement morgen	13
Abbildung 3	Klinische Pfade im Spannungsfeld zwischen Standardisierung und Flexibilisierung	14
Abbildung 4	Entkopplung von Prozessen von Anwendungen und Ressourcen	27
Abbildung 5	Überblick über die Konzepte in Zusammenhang mit Workflow Management (Quelle: WfMC 1999, S. 7)	28
Abbildung 6	Übersicht über die Struktur von WfMS (Quelle: Hollingsworth 1995, S. 13)	30
Abbildung 7	Überblick über die stationäre Behandlung von Wirbelsäulenerkrankungen	38
Abbildung 8	Taxonomie der Klassifikationsmerkmale für Varianz von klinischen Pfaden	46
Abbildung 9	Flexibilität durch Abstraktion von Prozessdetails	62
Abbildung 10	Selektion eines Worklets zur Realisierung einer Aktivität durch RDR-Regeln	63
Abbildung 11	Ad hoc Vervollständigung unspezifizierter Regionen der Prozessdefinition zur Laufzeit	66
Abbildung 12	Transformation von Prozessinstanzen in Abhängigkeit ihrer Ausführungshistorie	70
Abbildung 13	Notwendige Änderungsprimitive zum parallelen Einfügen einer neuen Aktivität	71
Abbildung 14	Abstraktion von Details einer Prozessänderung durch Nutzung von komplexen Änderungsoperationen	73
Abbildung 15	Prinzip der Blockstrukturierung in ADEPT2 [Reichert 2000b]	73

Abbildung 16	Überblick über Control Flow Pattern [Van der Aalst et al. 2003]	76
Abbildung 17	Nutzung von Meilensteinen zur Auswahl von Prozessvarianten	78
Abbildung 18	Festlegung von Ausführungsprioritäten zur Signalisierung von Abweichungen vom Standardablauf	79
Abbildung 19	Planbarer Vorwärtssprung mit Nachholen von übersprungenen Aktivitäten	79
Abbildung 20	Ableitung von Prozessvarianten vom Referenzprozessmodell und Suche an Hand von Teilprozessstrukturen	81
Abbildung 21	Konfiguration eines Prozessabschnittes zur Bild gebenden Diagnostik bei Vorliegen einer Schwangerschaft	83
Abbildung 22	Modellierung von Prozessvarianten als Optionen in Provop	85
Abbildung 23	SPOT-MM als Basis für Transformationen zwischen DSL und ausführbaren Prozessmodellierungssprachen	92
Abbildung 24	Modellierung von Behandlungsplänen mit Hilfe des SPOT-Care-Plan-Modelers	93
Abbildung 25	Repräsentation von Behandlungsplänen als ausführbare WSM Nets im Aristaflow® Template Editor	95
Abbildung 26	Ausführung von Behandlungsplänen mit Hilfe der Aristaflow® BPM Suite	96
Abbildung 27	Abgrenzung der kompositionalen Prozessmodellierung in SPOT vom traditionellen Vorgehen	98
Abbildung 28	Überblick über das Lösungskonzept	100
Abbildung 29	Problematik beim Management alternativer Prozessfragmente	104
Abbildung 30	Beispiel für die graphische Repräsentation eines Feature-Modells [Benavides et al. 2006]	106
Abbildung 31	Beispiel für die graphische Repräsentation eines kardinalitätsbasierten Feature-Modells	107
Abbildung 32	Bildung von SPFs durch die Zuordnung von elementaren Knoten des SPF-Typ-Graphen zu Prozessfragmenten	109
Abbildung 33	Bildung eines semantischen Kontexts durch die Knotenhierarchie des SPF-Typ-Graphen	110
Abbildung 34	Deklarative Prozessmodellierung auf Basis von SPF-Typ-Graphen	111
Abbildung 35	Zuordnung von logischen Verknüpfungen zu Knoten des SPF-Typ-Graphen	112
Abbildung 36	Knotenkardinalitäten im SPF-Typ-Graphen	113
Abbildung 37	Identische Teilgraphen innerhalb eines SPF-Typ-Graphen	114

Abbildung 38	Mehrfachreferenzen auf identische Teilgraphen innerhalb von SPF-Typ-Graphen	115
Abbildung 39	Constraint-Relationen in SPF-Typ-Graphen	116
Abbildung 40	Reduktion der Vielfalt möglicher Prozessdefinitionen durch semantische Anreicherung von SPF-Typ-Graphen	117
Abbildung 41	Gegenüberstellung von Knoten- und Attributdeklarationen in SPF-Typ-Graphen	118
Abbildung 42	Problematik inkonsistenter Datenflüsse durch Separation der Workflow Aspekte	119
Abbildung 43	Beispiele für Schnittstellenspezifikationen von Prozessfragmenten	121
Abbildung 44	Beispiele für Attributwerte von Parametern	123
Abbildung 45	Schnittstellenspezifikation für Prozessfragmente als UML Klassendiagramm	123
Abbildung 46	Metamodell für SPF-Typ-Graphen als UML Klassendiagramm	126
Abbildung 47	Beispiel für einen unerlaubten Zyklus in SPF-Typ-Graphen	130
Abbildung 48	Problematik mehrerer Constraint-Relationen innerhalb desselben Gültigkeitsbereichs	131
Abbildung 49	Beispiele für unerlaubte Kombinationen von logischen Operatoren und Constraint-Relationen	132
Abbildung 50	Vermeidung von Constraint-Relationen zwischen den direkten Nachfolgern desselben Kontextknotens	133
Abbildung 51	Problematik der Nicht-Selektierbarkeit von Knoten	133
Abbildung 52	Beispiel für die Ableitung von SPF-Graphen von einem gemeinsamen SPF-Typ-Graphen	137
Abbildung 53	Graphmorphismus als Bedingung für typisierte, vollständig konfigurierte SPF-Graphen	139
Abbildung 54	Auswirkung von Kardinalitäten auf die Struktur von SPF-Graphen	139
Abbildung 55	Auswirkung logischer Verknüpfungen auf die Struktur von SPF-Graphen	141
Abbildung 56	Auswirkung des Constraints »Anforderung« auf SPF-Graphen	142
Abbildung 57	Auswirkung des Constraints »Ausschluss« auf SPF-Graphen	143
Abbildung 58	Prinzip der Entwicklung von SPF-Graphen durch Anwendung von Produktionsregeln	144
Abbildung 59	Struktur und Funktionsweise von Produktionsregeln zur Manipulation von SPF-Graphen	146
Abbildung 60	Syntaktische Repräsentation zur kompakten Darstellung von Produktionsregeln	147
Abbildung 61	Umsetzung von Knotenkardinalitäten durch Produktionsregeln	148
Abbildung 62	Umsetzung des AND-Operators durch Produktionsregeln	149
Abbildung 63	Umsetzung des XOR-Operators durch Produktionsregeln	149

Abbildung 64	Umsetzung des OR-Operators durch Produktionsregeln	150
Abbildung 65	Umsetzung des OPT-Operators durch Produktionsregeln	150
Abbildung 66	Beispiel für eine Graphgrammatik	150
Abbildung 67	Beispiel für eine Constraint verträgliche Strukturierung von SPF-Typ-Graphen	157
Abbildung 68	Beispiel für eine Kontext erhaltende, Constraint verträgliche Strukturierung von SPF-Typ-Graphen	158
Abbildung 69	Unerwünschte Einschränkungen bei der Konfiguration von Klonen	160
Abbildung 70	Unvollständige und fehlerhafte Konfiguration in Zusammenhang mit dem Constraint »Anforderung«	162
Abbildung 71	Funktionsweise des Algorithmus zur nachträglichen Konfiguration eines SPF-Typ-Graphen	165
Abbildung 72	Funktionsweise des Algorithmus zur Rekonfiguration eines SPF-Typ-Graphen	167
Abbildung 73	Reduktion oder Rekonfiguration zur Durchsetzung von Ausschluss-Constraints in SPF-Graphen	169
Abbildung 74	Ausschnitt aus einem SPF-Typ-Graphen für chirurgische klinische Pfade	172
Abbildung 75	Optionen bei der Konfiguration von Klonknoten	176
Abbildung 76	Transformationsrelevante Kontrollflusskonstrukte von WSM Nets	181
Abbildung 77	Beispiel für ein einfaches WSM Net	182
Abbildung 78	Zusammenhang zwischen Änderungsprimitiven, komplexen Änderungsoperationen, höherwertigen Änderungsoperationen und Änderungstransaktionen in ADEPT2	185
Abbildung 79	Auswirkungen der ADEPT2-Änderungsoperation »serialInsert«	186
Abbildung 80	Auswirkungen der ADEPT2-Änderungsoperation »parallelInsert«	187
Abbildung 81	Auswirkung der ADEPT2 Änderungsoperation »insertBetweenNodeSets«	189
Abbildung 82	Mögliche Klon-Konstellationen im SPF-Graphen mit Relevanz für die Transformation	194
Abbildung 83	Initiale Struktur eines WSM Net in ADEPT2	195
Abbildung 84	Sequentielles Einfügen des ersten Prozessfragments	195
Abbildung 85	Paralleles Einfügen unabhängiger Prozessfragmente	196
Abbildung 86	Sequentielles Einfügen eines Prozessfragments vor Constraint-relevanten Aktivitäten (Beispiel 1)	197
Abbildung 87	Sequentielles Einfügen eines Prozessfragments vor Constraint-relevanten Aktivitäten (Beispiel 2)	198
Abbildung 88	Einfügen eines Prozessfragments parallel zu den Vorgängern der Constraint-relevanten Aktivitäten	199
Abbildung 89	Einfügen eines Prozessfragments parallel zu Constraint-relevanten Aktivitäten und deren Vorgänger	200

Abbildung 90	Serielles Einfügen eines neuen Prozessfragments zwischen zwei Aktivitätsmengen	201
Abbildung 91	Sequentielles Einfügen eines Prozessfragments zwischen zwei Aktivitätenmengen unter Berücksichtigung unabhängiger Aktivitäten	202
Abbildung 92	Einfügen eines Prozessfragments zwischen zwei Aktivitätenmenge und parallel zu unabhängigen Aktivitäten	203
Abbildung 93	Einfügen eines Prozessfragments zwischen zwei Aktivitätenmengen	204
Abbildung 94	Beispiel für ein WSM Net mit inkonsistentem Datenfluss	206
Abbildung 95	Beispiel für die Transformation eines SPF-Graphen unter Beachtung von Datenflussabhängigkeiten	207
Abbildung 96	Lösungsansatz zum Umgang mit Mehrfachreferenzen unter Vermeidung von Datenflussinkonsistenzen	209
Abbildung 97	Transformation eines nicht-terminalen Knotens in eine prozedurale, bedingte Verzweigungsstruktur	211
Abbildung 98	Umsetzung des Change Pattern zur nachträglichen Selektion von Prozessfragmenten	213
Abbildung 99	Umsetzung des Change Pattern zur nachträglichen Komposition von Prozessfragmenten	215
Abbildung 100	Umsetzung des Change Pattern zur mehrfachen Instanziierung eines Prozessfragments	216
Abbildung 101	Unterschied zwischen strukturerhaltender und nicht-strukturerhaltender Rekonfiguration von SPF-Graphen	224
Abbildung 102	Löschen von Prozessaktivitäten nach der Rekonfiguration des SPF-Graphen	228
Abbildung 103	Vereinfachtes Zustandsübergangsdiagramm für Instanzen von Prozessfragmenten in ADEPT2	231
Abbildung 104	Beispiel für eine Prozessinstanz auf Basis eines WSM Nets	233
Abbildung 105	Auswirkung des Ausschluss-Constraints auf das Einfügen von Knoten während der Rekonfiguration zur Laufzeit	236
Abbildung 106	Auswirkung des Anforderungs-Constraints auf das Einfügen von Knoten während der Rekonfiguration	237
Abbildung 107	Auswirkung des Anforderungs-Constraints beim Löschen eines Knotens während der Rekonfiguration	239
Abbildung 108	Einfügen von Prozessfragmenten in eine Prozessinstanz nach der Rekonfiguration des SPF-Graphen	243
Abbildung 109	Identifikation von Rekonfigurationsknoten in Abhängigkeit ihrer Knotenbezeichnung im SPF-Typ-Graphen	245
Abbildung 110	Kontextsensitive Identifikation von Rekonfigurationsknoten in Abhängigkeit ihrer Knotenbezeichnung	246

Abbildung 111	Kontextsensitive und prozessbezogene Identifikation von Knoten in Abhängigkeit ihrer Knotenbezeichnung	247
Abbildung 112	Beispiel für die Anwendung der SPF-Änderungsoperation INSERT auf einen SPF-Graphen	256
Abbildung 113	Unerwünschter Effekt beim Einfügen neuer Prozessfragmente in laufende Prozessinstanzen	257
Abbildung 114	Beispiel für die Anwendung der SPF-Änderungsoperation DELETE auf einen SPF-Graphen	259
Abbildung 115	Beispiel für die Anwendung der SPF-Änderungsoperation REPLACE auf einen SPF-Graphen	261
Abbildung 116	Beispiel für die Anwendung der SPF-Änderungsoperation RECONFIG auf einen SPF-Graphen	264
Abbildung 117	Anwendung einer Prozessvariante auf einen SPF-Graphen	266
Abbildung 118	Top-Level Prozess für den Aufruf von Prozessvarianten	267
Abbildung 119	Auswirkung des Korrektheitskriteriums für vollständige Sequenzialisierbarkeit auf die Struktur von SPF-Typ-Graphen	270
Abbildung 120	SPF-Typ-Graph für die Diagnostik von Wirbelsäulenerkrankungen	273
Abbildung 121	SPF-Typ-Graph für die Therapie von Wirbelsäulenerkrankungen	275
Abbildung 122	SPF-Graph für den klinischen Pfad zur Diagnostik von Wirbelsäulenerkrankungen	278
Abbildung 123	SPF-Graph für den klinischen Pfad zur Therapie von Wirbelsäulenerkrankungen	278
Abbildung 124	Transformation des »Knoten Anamnese«	279
Abbildung 125	Transformation des Knoten »Klinische Untersuchung«	280
Abbildung 126	Transformation des Knoten »Kleines Blutbild«	280
Abbildung 127	Transformation des Knoten »Röntgenuntersuchung«	280
Abbildung 128	Transformation des Knoten »MRT-Untersuchung«	281
Abbildung 129	Prozessdefinition für die Therapie von Wirbelsäulenerkrankungen nach der Transformation des SPF-Graphen	281
Abbildung 130	Instanz des klinischen Pfades zur Diagnostik von Wirbelsäulenerkrankungen	282
Abbildung 131	Instanz des Diagnostikprozesses nach der Rekonfiguration des SPF-Graphen und erneuter Transformation	283
Abbildung 132	Instanz des klinischen Pfades zur Therapie von Wirbelsäulenerkrankungen	283
Abbildung 133	Instanz des Therapieprozesses nach der Rekonfiguration des SPF-Graphen und erneuter Transformation	284

Abbildung 134	SPF-Typ-Graph der Prozessvariante »Diabetes mellitus« für die Diagnostik von Wirbelsäulenerkrankungen	285
Abbildung 135	SPF-Graph und Prozessdefinition der Prozessvariante »Diabetes mellitus« für die Diagnostik von Wirbelsäulenerkrankungen	285
Abbildung 136	Prozessinstanz zur Diagnostik von Wirbelsäulenerkrankungen nach Aufruf der Prozessvariante »Diabetes mellitus«	286
Abbildung 137	SPF-Typ-Graph der Prozessvariante »Diabetes mellitus« für die Therapie von Wirbelsäulenerkrankungen	286
Abbildung 138	SPF-Graph und Prozessdefinition der Prozessvariante »Diabetes mellitus« für die Therapie von Wirbelsäulenerkrankungen	287
Abbildung 139	Prozessdefinition zur Therapie von Wirbelsäulenerkrankungen nach Aufruf der Prozessvariante »Diabetes mellitus«	289
Abbildung 140	Übersicht über die Architektur des Prototypen	291
Abbildung 141	XML Schema für SPF-Typ-Graphen	293
Abbildung 142	SPF-Typ-Graph für die Behandlung von Wirbelsäulenerkrankungen im SPOT-Domain-Model-Configurator	295
Abbildung 143	Übersicht über die im SPF-Typ-Graphen definierten Constraint-Relationen	296
Abbildung 144	Konfiguration des Wurzelknotens im SPOT-Domain-Model-Configurator	297
Abbildung 145	Durchsetzung von Constraints und automatisierte Konfiguration von Knoten im SPOT-Domain-Model-Configurator	298
Abbildung 146	Auswahl von Klonknoten im SPOT-Domain-Model-Configurator	299
Abbildung 147	Gemeinsame Konfiguration von Klonknoten im SPOT-Domain-Model-Configurator	300
Abbildung 148	Vollständige Konfiguration des SPF-Typ-Graphen für die Behandlung von Wirbelsäulenerkrankungen im SPOT-Domain-Model-Configurator	301
Abbildung 149	Prozessinstanz des klinischen Pfades für die Diagnostik von Wirbelsäulenerkrankungen	302

A.4 Tabellenverzeichnis

Tabelle 1	Überblick über Definitionen für klinische Pfade	19
Tabelle 2	Kriterien zur Charakterisierung klinischer Pfade	21
Tabelle 3	Überblick über Definitionen für Pfadvarianz	44
Tabelle 4	Übersicht über die Anforderungen an die Umsetzung klinischer Pfade und den Umgang mit Varianz	54

Tabelle 5	Ergebnisse der Analyse von iMedOne.DocPath zur Entwicklung und Ausführung klinischer Pfade	58
Tabelle 6	Ergebnisse der Analyse von ClinPath zur Entwicklung und Ausführung klinischer Pfade	59
Tabelle 7	Ergebnisse der Analyse von Worklets zur Unterstützung manueller Abweichungsmaßnahmen	64
Tabelle 8	Ergebnisse der Analyse von Ansätzen des Late Modeling zur Unterstützung manueller Abweichungsmaßnahmen	67
Tabelle 9	Ergebnisse der Analyse von Declare zur Unterstützung manueller Abweichungsmaßnahmen	69
Tabelle 10	Ergebnisse der Analyse von CAKE2 und WASA2 zur Unterstützung manueller Abweichungsmaßnahmen	71
Tabelle 11	Ergebnisse der Analyse von ADEPT2 zur Unterstützung manueller Abweichungsmaßnahmen	74
Tabelle 12	Ergebnisse der Analyse von YAWL zur Unterstützung von Prozessvarianten	77
Tabelle 13	Ergebnisse der Analyse von ADEPT2 zur Unterstützung von Prozessvarianten	80
Tabelle 14	Ergebnisse der Analyse von C-YAWL und C-EPC zur Unterstützung von Prozessvarianten	83
Tabelle 15	Ergebnisse der Analyse von Provop zur Unterstützung von Prozessvarianten	85
Tabelle 16	Ergebnisse der Analyse von AGENTWORK zur Unterstützung von Prozessvarianten	88
Tabelle 17	Beispiele für Attributwerte von Methoden	121
Tabelle 18	Sequentielles Einfügen einer neuen Aktivität, Adaptation Pattern AP1 [Weber et al. 2008]	186
Tabelle 19	Paralleles Einfügen einer neuen Aktivität, Adaptation Pattern AP1 [Weber et al. 2008]	186
Tabelle 20	Setzen bzw. anpassen der Attribute von Prozessaktivitäten	188
Tabelle 21	Einfügen von Synchronisationskanten zwischen Prozessaktivitäten paralleler Zweige	188
Tabelle 22	Einfügen einer neuen Aktivität zwischen zwei Aktivitätenmengen	189
Tabelle 23	Entfernen einer Aktivität aus der Prozessdefinition, Change Pattern AP2 [Weber et al. 2008]	190
Tabelle 24	Entfernen eines vollständigen Kontrollblocks aus der Prozessdefinition, Change Pattern AP2 [Weber et al. 2008]	191
Tabelle 25	Definition der SPF-Änderungsoperation INSERT	254
Tabelle 26	Definition der SPF-Änderungsoperation DELETE	257
Tabelle 27	Definition der SPF-Änderungsoperation REPLACE	259
Tabelle 28	Definition der SPF-Änderungsoperation RECONFIG	262
Tabelle 29	Beispiele für Schnittstellenspezifikationen von Prozessfragmenten als Bestandteile von Prozessvarianten	268

Tabelle 30	Beispiel für die Spezifikation einer SPF-Änderungsoperation als Methode im Kontext von Prozessfragmenten	269
Tabelle 31	Spezifikation statischer Methodenparameter für die SPF-Änderungsoperation INSERT	269
Tabelle 32	Spezifikation dynamischer Methodenparameter für die SPF-Änderungsoperation INSERT	270
Tabelle 33	Schnittstellenspezifikation für das Prozessfragment »Anamnesebogen ausfüllen«	275
Tabelle 34	Ergebnisse der Analyse des Lösungskonzeptes zur Entwicklung und Ausführung klinischer Pfade	305
Tabelle 35	Ergebnisse der Analyse des Lösungskonzeptes zur Unterstützung manueller Abweichungsmaßnahmen	307
Tabelle 36	Ergebnisse der Analyse des Lösungskonzeptes zur Unterstützung von Prozessvarianten	310

A.5 Formelverzeichnis

Formel 1	Hauptalgorithmus zur rekursiven Erzeugung von Produktionsregeln	151
Formel 2	Algorithmus zur Erzeugung von Produktionsregeln entsprechend der Kardinalität eines Knoten	152
Formel 3	Algorithmus zur Erzeugung von Produktionsregeln für Knoten mit AND-Operator	153
Formel 4	Algorithmus zur Erzeugung von Produktionsregeln für Knoten mit XOR-Operator	153
Formel 5	Algorithmus zur Erzeugung von Produktionsregeln für Knoten mit OR-Operator	153
Formel 6	Algorithmus zur Erzeugung von Produktionsregeln für Knoten mit OPT-Operator	154
Formel 7	Initialisierung von SPF-Graphen	154
Formel 8	Algorithmus zur Transformation von SPF-Graphen durch Regelanwendung	154
Formel 9	Algorithmus zur Überprüfung der Einhaltung der Constraint-Relationen »Anforderung« und »Ausschluss«	160
Formel 10	Algorithmus zur Identifikation von Knoten gemäß Constraintbedingung	161
Formel 11	Hauptalgorithmus zur Durchsetzung von Anforderungs-Constraints	163
Formel 12	Algorithmus zur nachträglichen Konfiguration des SPF-Typ-Graphen zur Durchsetzung von Anforderungs-Constraints	165
Formel 13	Algorithmus zur Rekonfiguration des SPF-Typ-Graphen zur Durchsetzung von Anforderungsconstraints	167
Formel 14	Algorithmus zur Durchsetzung von Ausschluss-Constraints	169

Formel 15	Algorithmus für die automatisierte Konfiguration	174
Formel 16	Gleichförmige Konfiguration von Klonknoten	177
Formel 17	Pseudocode für die neue höherwertige Änderungsoperation »insertBeforeNodeSet«	190
Formel 18	Algorithmus zur nicht-strukturerhaltenden Rekonfiguration einer Knotenmenge zur Modellierungszeit	224
Formel 19	Algorithmus für das strukturerhaltende Einfügen nicht-terminaler Knoten in SPF-Graphen zur Modellierungszeit	225
Formel 20	Algorithmus für das strukturerhaltende Entfernen von Teilgraphen aus SPF-Graphen zur Modellierungszeit	226
Formel 21	Algorithmus zur nicht-strukturerhaltenden Rekonfiguration einer Knotenmenge zur Laufzeit	239
Formel 22	Algorithmus für das strukturerhaltende Einfügen nicht-terminaler Knoten in SPF-Graphen zur Laufzeit	240
Formel 23	Algorithmus für das strukturerhaltende Entfernen von Teilgraphen aus SPF-Graphen zur Laufzeit	241
Formel 24	Algorithmus zur automatischen Identifikation von Knoten im SPF-Graphen	248
Formel 25	Algorithmus zur Umsetzung der atomaren Änderungsoperation INSERT	254
Formel 26	Algorithmus zur Umsetzung der SPF-Änderungsoperation DELETE	258
Formel 27	Algorithmus zur Umsetzung der SPF-Änderungsoperation REPLACE	260
Formel 28	Algorithmus zur Umsetzung der SPF-Änderungsoperation RECONFIG	262
Formel 29	Graphgrammatik für den SPF-Graph zur Diagnostik von Wirbelsäulenerkrankungen	276
Formel 30	XML-Schema für SPF-Typ-Graphen	341
Formel 31	XML-Repräsentation des SPF-Graphen für den klinischen Pfad zur Diagnostik von Wirbelsäulenerkrankungen	343

A.6 Abkürzungsverzeichnis

Abkürzung	Bedeutung
ADEPT	Application Development Based on Pre-modeled Encapsulated Process Templates
API	Application Programming Interface
AWMF	Arbeitsgemeinschaft der wissenschaftlichen medizinischen Fachgesellschaften
ÄZQ	Ärztliches Zentrum für Qualität
BÄK	Bundesärztekammer
BPM	Business Process Management
BPMN	Business Process Modeling Notation
CAKE	Collaborative Agent-based Knowledge Engine
C-EPC	Configurable Event Process Chain
CT	Computertomographie

C-YAWL	Configurable YAWL
DiPP	Digitale Pfade im Gesundheitsnetz Prosper
DRG	Diagnosis Related Groups
DSL	Domain-specific Language
ECA	Event Condition Action
EPC	Event Process Chain
ERP	Enterprise Resource Planning System
FODA	Feature-Oriented Domain Analysis
FORM	Feature-Oriented Reuse Method
FPG	Flexible Prozessgraphen
GG	Graphgrammatik
HL7 RIM	Health Level 7 Reference Information Model
IAO	Institut für Arbeitswirtschaft und Organisation
ICD	International Classification of Diseases
IML	Institut für Materialfluss und Logistik
ISST	Institut für Software- und Systemtechnik
Jar	Java archive
KBS	Knappschaft-Bahn-See
KBV	Kassenärztliche Bundesvereinigung
KIS	Krankenhausinformationssystem
LOINC	Logical Observation Identifiers Names and Codes
LTL	Lineare Temporale Logik
MDA	Model-driven Architecture
MRT	Magnetresonanztomographie
OCL	Object Constraint Language
OPM	Object Process Methodology
OPS	Operationen- und Prozedurenschlüssel
PESOA	Process Family Engineering in Service-Oriented Architectures
PMC	Patient Management Categories
PoF	Pockets of Flexibility
PR	Produktionsregel
Provop	Prozessvarianten mittels Optionen
PV	Prozessvariante
RDR	Ripple-down Rules
SAT	Satisfiability problem
SG	SPF-Graph
Snomed CT	Systemized Nomenclature for Medicine-Clinical Terms
SOA	Serviceorientierte Architektur
SPF	Semantisches Prozessfragment
SPOT	Servicebasierte und prozessorientierte Orchestrierungstechnologie
SPOT-ML	SPOT Modeling Language
SPOT-MM	SPOT Metamodell
STG	SPF-Typ-Graph
UML	Unified Modeling Language
WASA	Workflow-based Architecture to support Scientific Applications
WfMC	Workflow Management Coalition
WfMS	Workflow Management System
WIDE	Workflow on Intelligent Distributed database Environment
WS-BPEL	Web Service Business Process Execution Language
WSDL	Web Service Description Language
WSM Net	Well-structured Markup Net
XML	Extensible Markup Language
XPATH	XML Path Language
XSLT	Extensible Stylesheet Language Transformations
YAWL	Yet Another Workflow Language