Feature-orientierte Entwicklung von rollenbasierten Systemen zur kooperativen Entscheidungsfindung

Dissertation

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik
von

Martin Karusseit

Dortmund

2009

Tag der mündlichen Prüfung:	15.12.2009
Dekan:	Prof. Dr. Peter Buchholz
Gutachter:	Prof. Dr. Bernhard Steffen
	Prof. Dr. Dietmar Jannach

Danksagung

Ich möchte mich besonders bei meinem Doktorvater Prof. Dr. Bernhard Steffen und Prof. Dr. Ing. Tiziana Margaria bedanken, die mir nach meinem Studium der Elektrotechnik und mehrjähriger Tätigkeit in der Wirtschaft den Weg zurück an die Technische Universität Dortmund aufgezeigt haben. Durch ihr weitreichendes und fundiertes Wissen in ihren Fachgebieten haben sie mich während meiner Promotion unterstützend begleitet. Die fachlichen Diskussionen und Anregungen waren mir immer eine große Hilfe.

Des Weiteren möchte ich mich bei Prof. Dr. Dietmar Jannach bedanken, der sich bereit erklärt hat, Zweitgutachter für meine Arbeit zu sein und mir hilfreiche Anregungen in der Fertigstellungsphase der Dissertation gegeben hat.

Weiterer Dank gilt Prof. Dr. Kern-Isberner für die Leitung meiner Prüfungskommission und Dr. Oliver Rüthing, der einerseits das wissenschaftliche Mitglied in der Prüfungskommission ist und sich andererseits für hilfreiche fachliche Gespräche Zeit genommen hat.

Die Umsetzung eines so großen Projektes, wie es der OCS mit seiner Dienstfamilie darstellt, ist nur mit der Unterstützung eines motivierten Teams zu bewältigen. Ich möchte mich daher bei meinem ehemaligen OCS-Team Holger Willebrandt, Markus Korte, Marc von Renteln, Eugen Reinke, Sebastian Hundt und Dominik Brink für die jahrelange erfolgreiche Zusammenarbeit bedanken. Des Weiteren bedanke ich mich bei Markus Bajohr, der mit seinem fundierten Wissen im Bereich der Netzwerktechnik, die Umsetzung der OCS-Clusters durchgeführt hat. Einen besonderen Dank auch an Ben Lindner für die sehr gute Zusammenarbeit zu Beginn des OCS Projektes.

Ich möchte mich ebenfalls bei dem gesamten Team des Lehrstuhls für Programmiersysteme der Technischen Universität Dortmund für die angenehme Arbeitsatmosphäre bedanken.

Danke an Springer Heidelberg, insbesondere an Herrn Alfred Hofmann, Frank Holzwarth und Ursula Barth, die ein sehr angenehmes Arbeitsklima seitens

des Projektpartners geschaffen haben.

Ich bedanke mich bei Bettina, Claudia, Oliver, Thomas und Volker für das Auffinden des Fehlerteufels, der gerade während der zahlreichen Nachtarbeiten einen leichten Einzug in die Vorversionen dieser Arbeit gefunden hatte.

Abschließend möchte ich mich für die unermessliche Unterstützung bei meiner Frau Michaela, meinem Sohn Tobias und meiner Tochter Stella herzlich bedanken. Sie haben oftmals auf mich verzichten müssen und haben mich in schwierigen Situationen mit ihrer großen Liebe unterstützt und Kraft gegeben.

Zusammenfassung

Effiziente Entscheidungsfindungen spielen eine wirtschaftlich bedeutende Rolle in Bezug auf den Arbeitsaufwand und die entstehenden Kosten. Mit steigender Anzahl der Entscheidungsträger treten terminliche und organisatorische Engpässe auf, die durch die globale Verteilung der Personen und die daraus resultierende Zeitzonenproblematik verstärkt werden. Im wissenschaftlichen und industriellen Bereich ist die Verwaltung und Koordination der Begutachtungsprozesse von Artikeln für die Leitung und die Beteiligten des Prozesses mit einem enormen zeitlichen Aufwand verbunden. Das OCS [1]-Projekt realisiert web-basierte Begutachtungssysteme, welche die (wiederkehrenden) Prozesse automatisieren und die relevanten Daten und Ergebnisse unabhängig von Zeit und Ort transparent zur kooperativen Zusammenarbeit zur Verfügung stellen.

In dieser Arbeit wird ein auf der Modellgetriebenen Softwareentwicklung (MDSD [2]) basierende Feature-orientierte Vorgehenweise für die Softwareentwicklung und ein darauf aufbauender Rollen-basierter Zugriffskontrollmechanismus im Rahmen einer konkreten praktischen Umsetzung vorgestellt. Die auf Basis des ABC [3] konzipierten graphbasierten Ansätze verfolgten bereits Ende der Neunzigerjahre eine Service-orientierte Softwareentwicklung und zeigten eine Lösung für die Umsetzung komplexer Anwendungen für ausführbare Geschäftsprozesse auf, die erst später, etwa durch die im Jahre 2002 eingeführte Sprache für Geschäftsprozesse WS-BPEL, in den Fokus intensiver Forschung und einer breiten öffentlichen Beachtung gelangten. Das Feature-orientierte Konzept ist technologieunabhängig und wurde im Rahmen des OCS-Projekts erprobt. Die Begutachtungssysteme finden ihren Einsatz im internationalen wissenschaftlichen Bereich von Konferenzen, Journals und Workshops. Dort werden sie eingesetzt, um den gesamten Begutachtungsprozess für wissenschaftliche Artikel abzubilden. Die Begutachtungssysteme sind web-basierte Applikationen, die alle relevanten Daten, z.B. einer Konferenz, online perso-

[1] Online Conference Service

[2] Model-Driven Software Development

[3] Agent Building Center: Eine graphbasierte Entwicklungsumgebung

nalisiert zur Verfügung stellen. Durch die Gruppierung der Dienstfunktionalitäten in Features und die Modellierung derselben als Graphen, wird eine hohe Agilität des Systems in Bezug auf neue Kundenanforderungen erreicht, was sich in der bislang zehnjährigen Erfahrung bewährt hat. Mittels der Verifikation der Graphmodelle des OCS, die den Kontrollfluß (Geschäftslogik) repräsentieren, wird gewährleistet, dass neu erstellte bzw. modifizierte Graphmodelle stets konsistent zu den ursprünglichen Anforderungen bleiben. Das entwickelte Benutzer/Rollen/Rechte-Modell verleiht den Systemen eine Flexibilität bzgl. der Zugriffsrechte und Personalisierung der Dienstfunktionalitäten. Der erfolgreiche Einsatz der Konzepte hat sich durch die bis heute jahrelange Verwendung der realisierten Begutachtungssysteme *OCS*, *OJS* [4], *LNCS* [5] und *Transactions* bestätigt.

[4]Online Journal Service

[5]Lecture Notes in Computer Science

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Teil I

Motivation und Hintergrund

Kapitel 1

Einleitung

1.1 Motivation

Effiziente Entscheidungsfindungen sind in allen Bereichen der Gesellschaft und der Industrie von höchster Bedeutung. Entscheidungen werden in der Regel von mehreren Personen bzw. Gruppen gemeinsam getroffen, was in Abhängigkeit von der Anzahl der Involvierten eine steigende Komplexität der Terminfindung und der Koordination zufolge hat. Die Aspekte der globalen Verteilung der beteiligten Personen, die dadurch auftretenden Zeitzonen und die verschiedenartigen Arbeitscharaktere bilden weitere Randbedingungen für die Komplexität einer gemeinsamen Entscheidungsfindung und einer daraus resultierenden Verzögerung von Fristen.

Internetbasierte Software-Anwendungen sind prädestiniert für die systemgestützte Steuerung derartiger kooperativer Geschäftsprozesse, wie Kommissions-, Komitee-, oder Ausschusssitzungen. Sie vereinigen wichtige Merkmale in Bezug auf eine unterstützende Entscheidungsfindung und Echtzeitverhalten. Insbesondere die Bereitstellung der zugrunde liegenden Informationen unabhängig von Zeit und Ort (24x7 Verfügbarkeit) und die Bedienbarkeit mittels eines gängigen Internet-Browsers sind hier hervorzuheben. Arbeitsaufteilungen und deren Zeitplanung, Kommunikationskomponenten und Konfliktbehandlungen können in die Web-Anwendung integriert, und neu entwickelte Funktionalitäten dem Benutzer auf einfache Weise personalisiert zugänglich gemacht werden. Daten und Ergebnisse sind vom Benutzer zu jeder Zeit zentralisiert editierbar. Derartige CSCW (Computer Supported Cooperative Work) An-

wendungen dienen der Automatisierung von (wiederkehrenden) Abläufen, was zu einer effizienteren und insbesondere Kosten sparenden Abwicklung von Geschäftsprozessen führt. Es ist nicht mehr erforderlich, die zu diskutierenden Dokumente aufwendig per Post oder Email zu versenden und zu verwalten. Die Reisekosten für Treffen können auf wenige Termine reduziert und der eigene Zeit- und Arbeitsaufwand minimiert werden. Mittels der "virtuellen" Konferenz werden die Aspekte der Vorbereitungsphase einer "physikalisch" stattfindenden Konferenz abgedeckt.

Einen klassischen Anwendungsfall stellen Begutachtungssysteme (peer review systems) für Dokumente und Papiere im wissenschaftlichen Umfeld dar. Der Umfang einer solchen Begutachtung setzt sich im Durchschnitt aus 150 zu begutachtenden Papieren zusammen, die an ein Komitee von 20 - 30 Teilnehmern homogen zur Begutachtung zu verteilen sind, wobei pro Beitrag mindestens drei Gutachter aus dem Komitee ausgewählt werden. Der resultierende Verwaltungs- und Koordinationsaufwand stellt für die Leiter eine große und insbesondere für eine einzelne Person nur schwer lösbare Herausforderung dar. So ist es von größter Wichtigkeit, im Vorfeld einer Konferenz eine gute Basis für den Begutachtungsprozess zu schaffen, um eine adäquate Auswahl an Dokumenten und somit eine Qualitätssicherung der Konferenz zu gewährleisten.

Ende der neunziger Jahre hielten die ersten wenigen internetbasierten Begutachtungssysteme ihren Einzug in die Wissenschaftsgemeinde und revolutionierten das Arbeiten für Organisatoren und Teilnehmer von wissenschaftlichen Konferenzen. Dienten diese Systeme damals zur Abbildung der eigentlichen Begutachtungsprozesse, unterstützen sie heutzutage den gesamten Ablauf einer Konferenz (siehe Kapitel 1.2).

Der Online Conference Service (OCS [1]) [LMS01, MK02, DDB+04, HBTE+05, KM05, KM06, KMW08, OCS08], dessen Konzepte und Entwicklung den Kern dieser Arbeit darstellen, reiht sich mit in die Gruppe der ersten Systeme ein, die im Bereich der Begutachtung von wissenschaftlichen Arbeiten eine Vorreiterstellung einnehmen. OCS war eines der ersten *State of the Art* Systeme für Begutachtungsprozesse, das den Begutachtungsworkflow mitbestimmt und andere Begutachtungssysteme in ihrer Realisierung beeinflusst hat. Mit der Konzeption und Umsetzung des OCS war ich Ende der Neunzigerjahre in die Entstehungszeit der ersten Begutachtungssystemen involviert und hatte einen direkten Bezug zum Kunden. Die umfassende Vertrautheit der Praxis, die ich durch die Projektleitung, Programmierung, Serverbetreuung der Live-Systeme

[1] Die Leitung des OCS-Projektes wird am Lehrstuhl für Programmiersysteme an der TU Dortmund durch Prof. Dr. B. Steffen und den Verfasser dieser Arbeit, Dipl. Ing. M. Karusseit, durchgeführt.

und den Support der Dienstbenutzer erlangen konnte, haben die realisierten Begutachtungssysteme, insbesondere den OCS, über die Jahre zu etablierten Systemen herangewachsen lassen. Der OCS wird von Springer Heidelberg als Begutachtungssystem für die LNCS-Reihe eingesetzt und erfüllt im Produktionsbetrieb höchste Anforderungen an Stabilität.

OCS ist eine web-basierte Anwendung, die alle nötigen Geschäftsprozesse und Funktionalitäten eines Online-Entscheidungssystems in sich vereint. OCS bildet eine Dienstfamilie mit dem Online Journal Service (OJS) [HBTE+05], dem Management Overview Service (MOS), Proceeding Production Service (PPS) [Hol06], den Lecture Notes in Computer Science (LNCS) Proposal Service, und den LNCS Transactions (siehe Kapitel 7).

Der OCS blickt nun auf eine zehnjährige Geschichte zurück, in der er erfolgreich für die Vorbereitung von Konferenzen und Journals eingesetzt wurde. Durch die stetig wachsende Anzahl von mehr als 30.000 Benutzern und den daraus resultierenden und teilweise implementierten Änderungs- und Erweiterungswünschen kann sich der OCS an ein weites Spektrum von Begutachtungsprozessen verschiedener Kommunities anpassen.

Ich arbeite seit 1999 mit dem Springer Verlag Heidelberg [Hei09c] als Projektpartner zusammen, der die Dienste für die bei LNCS [Hei09d] publizierenden Tagungen einsetzt. Seit 2003 bietet Springer den OCS offiziell als Konferenzsystem an [Hei09b], das von 2007 an von Springer SBM [2] in Dordrecht, Niederlande, gehostet wird.

Web-basierte Applikationen wie der OCS sind sehr komplexe Systeme, die auf einer verteilten mehrschichtigen Softwarearchitektur beruhen (siehe Kapitel 6.1 und 6.2). Aufgrund der stetig wachsenden Komplexität der Systeme ist es von großem Interesse, ein adäquates Entwicklungskonzept zu verwenden, um Änderungen oder Integrationen einerseits schnell und unkompliziert durchführen und andererseits das geänderte System auf dessen Funktion validieren zu können. Des Weiteren verwalten Begutachtungssysteme sensible Daten, auf die mehrere Hundert Benutzer pro Konferenz in Abhängigkeit ihrer Rechte Zugriff erhalten. Es ist daher von zentraler Bedeutung, einen geeigneten Zugriffskontrollmechanismus zu verwenden, der unautorisierte Zugriffe verhindert.

Der OCS ist ein Feature-basiertes System, das modular und komponentenbasiert aufgebaut ist. Der Kontrollfluss (Geschäftslogik) des OCS ist als gerichtete Graphen modelliert, wobei die Knoten die Dienstfunktionalitäten oder Dienste selbst und die Kanten die Übergänge von Knoten zu Knoten darstel-

[2] Springer Science+Business Media

len. Diese Graphen können bereits zur Designzeit einer Applikation mittels eines Modelcheckers auf zentrale Eigenschaften hin überprüft werden. Hierfür werden atomare Eigenschaften (Propositionen) an die Knoten der Graphen annotiert. Die Verifikation der Systembeschreibung (Graphmodell) wird gegen eine Spezifikation durchgeführt. Diese Spezifikation wird als temporal logische Formel in CTL (Computation Tree Logic) geschrieben, die auf den annotierten Propositionen aufbaut. Der Modelchecker überprüft diese Grapheigenschaften und zeigt vorhandene Inkonsistenzen auf. Ein einfaches Beispiel ist die Überprüfung des Zugriffs auf Seiten des internen Bereichs einer Applikation, der nur mit einer erfolgreichen Authentifizierung zugelassen werden darf.

Diese Arbeit stellt ein neuartiges Konzept in der Entwicklung von komplexen, Web-basierten Entscheidungssystemen vor, das bereits frühzeitig den Service-orientierten und modellgetriebenen Gedanken in sich vereinte. Die einheitliche Vorgehensweise verfolgt neben der vereinfachten Entwicklung von komplexen Anwendungen, in der ein Anwendungsexperte von Beginn an involviert wird, die kontinuierliche Weiterentwicklung [Web99] der Anwendung. Die zentralen Unterschiede zu anderen Entscheidungssystemen sind

- die **Agilität** und **Anpassbarkeit** in Hinsicht auf die Modifikation eines bestehenden Dienstes,
- die einfache **Erweiterbarkeit** durch einen modularen graphbasierten Ansatz und
- die **Zuverlässigkeit** und **Sicherheit** des Gesamtsystems durch den Einsatz von Model Checking [Ste91, MOSS99] und eines eigenen Zugriffskontrollmechanismus, der ebenfalls mittels Model Checking verifizierbar ist.

Diese Eigenschaften sind für die Softwareentwicklung von großer Bedeutung. Softwaresysteme müssen auf neue Kundenanforderungen schnell und sicher reagieren können. Schnell in der Hinsicht, dass ein Entwickler die entsprechenden Stellen im Quellcode findet und sicher in der Art, dass die Modifikationen lauffähig sind und keine bestehenden Dienstfunktionalitäten negativ beeinflussen. Sind Anfragen zu sensiblen Daten zu bedienen, wie es bei Entscheidungssystemen vorrangig der Fall ist, ist es um so wichtiger, den gesicherten Zugriff nach einer Modifikation zu gewährleisten. So wäre es zum Beispiel ein extremes Problem, wenn nach einer Änderung die Autoren Zugriff auf die nicht anonymisierten Gutachten erhalten und somit die Namen derer erfahren, die ihren Artikel beurteilt haben.

Des Weiteren vergleicht diese Arbeit die Konzepte mit anderen Arbeiten aus dem technischen und wissenschaftlichen Themengebiet (siehe Kapitel 2). Der Fokus wird auf das Begutachtungssystem OCS und dessen Dienstfamilie gelegt, es werden insbesondere der technologieunabhängige Feature-orientierte Entwicklungsansatz und ein darauf basierender flexibler, rollenbasierter Zugriffskontrollmechanismus erörtert. Weiterführend werden Analysen, Erfahrungen und Fallstudien aufgezeigt, die durch den Einsatz des OCS erforscht wurden. Eine detailliertere Beschreibung der Inhalte und der Struktur dieser Arbeit werden in den Kapiteln 1.3 und 1.4 aufgezeigt.

1.2 Entscheidungssystem

Ein Entscheidungssystem bildet Modelle zur Entscheidungsfindung ab, die in Form eines Softwaresystems den Entscheidungsprozess unterstützen. Ein Einsatzgebiet von Entscheidungssystemen ist die Begutachtung von wissenschaftlichen Arbeiten. Begutachtungssysteme und deren Prozesse unterliegen einer Reihe von Kriterien, Randbedingungen und Anforderungen, die im Folgenden näher betrachet werden.

1.2.1 Begutachtungsprozess

Der generelle Ablauf einer Begutachtung für Artikel [3] besteht aus den folgenden vier Hauptphasen:

1. Einreichungsphase
2. Verteilungsphase der zu begutachtenden Artikel
3. Begutachtungsphase
4. Diskussions- und Entscheidungsphase

Ziel eines solchen Prozesses ist es, aus einer Menge von Artikeln unter Berücksichtigung verschiedener Gesichtspunkte adäquate Artikel zu selektieren. Die angenommenen Artikel werden von den Autoren auf einer Konferenz vorge-

[3] Ein Artikel bezeichnet im wissenschaftlichen Umfeld das Ergebnis einer Forschungsarbeit oder Fallstudie in schriftlicher Form.

tragen und in Form eines Tagungsbandes [4] oder Journals [5] veröffentlicht. Die Entscheidung über Akzeptanz oder Ablehnung eines Artikels trifft das Programmkomitee auf Basis von Gutachten. Das Programmkomitee setzt sich aus dem Komiteeleiter und den Gutachtern zusammen. Die Kriterien der verschiedenen Kommunities für die Akzeptanz eines Artikels stimmen nicht immer vollständig überein. Zu den gemeinsamen Kriterien zählen:

- die Einhaltung von Fristen, z.B. die der Einreichungsfrist,
- die Verwendung des vorgegebenen Formats (Stilvorgabe und Seitenzahl des Artikels),
- die Übereinstimmung mit dem Fokus der Konferenz,
- und der innovative Anspruch an den Artikel.

Die wesentlichen Unterschiede liegen in den für die Entscheidung zugrunde liegenden Informationen, die in Form von Gutachten von den Komiteemitgliedern verfasst werden. Die Vorlagen für die Gutachten weisen in den einzelnen Kommunities unterschiedliche Fragestellungen und Auswertungsfunktionen auf.

Konkreter Anwendungsfall

Es gibt eine Reihe von Anwendungsfällen die einen Begutachtungsprozess widerspiegeln. Im Folgenden wird ein konkreter in der Praxis mehrfach bewährter Prozess einer Begutachtung am Anwendungsbeispiel des OCS vorgestellt.

Im OCS gibt es vordefinierte Rollen, die unterschiedliche Rechte und Sichten auf die Features des Dienstes definieren. Die Namen der Rollen und deren Funktionen sind in Tabelle 1.1 beschrieben.

Der OCS verfügt über ein rekonfigurierbares Rollen/Rechte-System, das die Modifikation der vordefinierten Rollen erlaubt und den PC Chair in die Lage versetzt, neue Rollen zu definieren. Durch diese Flexibilität kann der OCS speziell auf Kommunities mit abweichenden Rollen angepasst werden.

Abbildung 1.1 zeigt vereinfacht den Haupt-Begutachtungsprozess des OCS und die in ihm involvierten Rollen. Der Begutachtungsprozess startet bei Konferenzen mit dem *Call for Papers*. Der Aufruf beinhaltet die nötigen Daten wie Konferenzname, Homepage, Themengebiete, Deadlines, Bedingungen, Tagungsort

[4] Ein Tagungsband beinhaltet Informationen über die Konferenz, das Fachgebiet, die Organisatoren und die Artikel.

[5] Journals sind Zeitschriften die eine Auswahl ("special sections") von bereits veröffentlichten Artikeln in einem Fachgebiet veröffentlichen.

Author	Reicht ein oder mehrere Artikel zur Konferenz ein.
CoAuthor	Ist Mitverfasser eines Artikels.
PC Member	Ist für die Begutachtung von Artikeln zuständig und bestimmt mit, welche Artikel angenommen bzw. abgelehnt werden.
SubReviewer	Eine optionale Rolle, die den PC Member bei der Erstellung von Gutachten unterstützt.
PC Chair	Diese Rolle leitet und koordiniert die Computerunterstützte Konferenz.
Administrator	Übernimmt die technische Verwaltung des Dienstes.

Tabelle 1.1: Vordefinierte Rollen im OCS

und die Kontaktdaten der Organisatoren. Autoren werden dazu aufgefordert, Forschungsergebnisse, Tool-Präsentationen oder Fallstudien in vorgegebener schriftlicher Form einzureichen.

Nach einer erfolgreichen Registrierung über den OCS können die Autoren ihren Artikel elektronisch in das Entscheidungssystem hochladen. Der Dienst stellt hierfür ein Einreichungsformular zur Verfügung, das die Metadaten und eine druckbare elektronische Version (überwiegend PDF) des Artikels verlangt. Zu den Metadaten gehören in erster Linie der Titel des Artikels, Namen und Kontaktdaten aller Autoren, das Themengebiet und eine Zusammenfassung des Artikelinhaltes (Abstract). Optional kann dieser **Einreichungphase** des Artikels die Einreichung von Abstracts vorgeschaltet werden, die nur die Metadaten benötigt. Dieses dient den Organisatoren der Konferenz dazu, frühzeitig einen ersten Überblick über die potentiellen Autoren und deren Einreichungen zu erlangen. Insbesondere die Zuordnung der Artikel zu den Gutachtern kann bereits in der Einreichungsphase geschehen und eventuelle Lücken an Fachkompetenz durch Aquirierung von zusätzlichen Gutachtern ausgeglichen werden (vgl. Schritt 1 “submit article” in Abb. 1.1).

Nach Beendigung der Einreichungsphase beginnt die **Verteilungsphase** für den Leiter der Konferenz, dem PC Chair (vgl. Schritt 2 “delegate article” in Abb. 1.1). Er erhält zu jeder Einreichung vom Dienst automatisch eine Benachrichtigung via Email zugesandt und kann direkt auf den neuen Artikel reagieren. Die Verteilung der Artikel an die PC Member kann entweder manuell geschehen oder unter Verwendung der automatisierten Verteilungskomponente des OCS (siehe Kapitel 4.3). Bei einer durchschnittlichen Einreichungsanzahl von 150 Artikeln und einer Begutachteranzahl von drei pro Artikel, ergeben sich 450 Verteilungen. Zum Beispiel hatte die Konferenz *Tools and Algorithms*

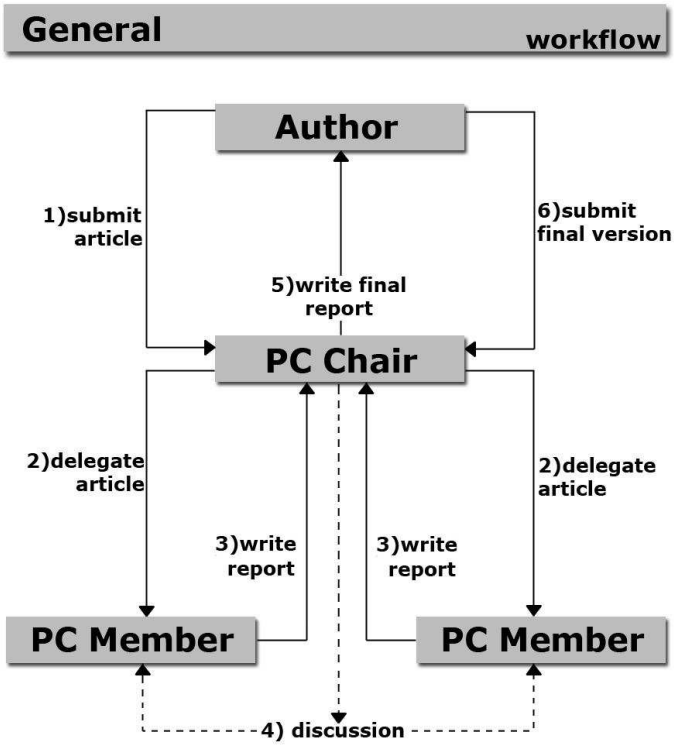


Abbildung 1.1: Vereinfachter OCS-Begutachtungsprozess

for the Construction and Analysis of Systems 2009 (TACAS 2009) [6] eine Anzahl von 131 Artikeln und eine daraus resultierende Verteilungszahl von 441, bedingt durch die Verteilung von mehreren Artikeln an vier Gutachter. Die in China stattfindende Konferenz *Advanced Data Mining And Applications 2009* (ADMA) gehört mit 322 Artikeln, 1169 Verteilungen und 795 Benutzern zu den größeren Konferenzen.

Die **Begutachtungsphase** (vgl. Schritt 3 "write report" in Abb. 1.1) beginnt, sobald die Artikel verteilt werden und die entsprechenden PC Members eine Benachrichtigung via Email vom OCS erhalten. Die PC Member sehen sich die ihnen zugewiesenen Artikel an und entscheiden, ob sie diese begutachten wollen oder nicht. Ein Grund der Ablehnung kann die fehlende Fachkompetenz in der Thematik des Artikels oder der wissenschaftliche Kontakt mit den Autoren sein, was unter Umständen einen Einfluss auf das Ergebnis des Gutachtens

[6] TACAS 2009 ist eine Hauptkonferenz der *European Joint Conferences on Theory and Practice of Software* (ETAPS)

zufolge hätte. Im Falle der Ablehnung informiert der OCS den PC Chair und bietet den Artikel zur erneuten Zuweisung an. Optional kann der PC Member seine zugewiesenen Artikel an SubReviewer weiterleiten. Mittels der Gutachten der SubReviewer schreibt der PC Member anschließend sein eigenes Gutachten. Dieses ist ein verbreiteter Vorgang, da PC Members unter Umständen in mehreren Programmkomitees involviert sind und im Durchschnitt pro Konferenz 10-20 Artikel begutachten müssen. Aufgrund das zeitgleich viele Gutachten zu schreiben sind, ist es auch im Sinne des PC Chairs, dass der PC Member bei fehlender Kompetenz selber einen adäquaten SubReviewer mit entsprechender Facherfahrung sucht und in den Begutachtungsprozess involviert.

Nachdem die Gutachten eingegangen sind, eröffnet der PC Chair die **Diskussions**- und **Entscheidungsphase**, in der über die Auswahl der Artikel entschieden wird. OCS stellt für jeden Artikel ein Diskussionsforum zur Verfügung, auf das die PC Member in Abhängigkeit ihrer Themengebiete Zugriff haben. Die Teilnehmer eines Diskussionsforums erhalten bei jeder neuen Nachricht im Forum eine Mitteilung via Email. Die Email enthält für den Teilnehmer ein Ticket für den direkten Zugang zum Forum. Das Ticket ist nur einmalig verwendbar und bietet einen schnellen und direkten Zugriff. Die Diskussion (vgl. Schritt 4 “discussion” in Abb. 1.1) über die Annahme oder Ablehnung eines Artikels lässt sich unterteilen in die lokale und globale Diskussion:

- **Lokale Diskussion:** Die PC Members, die einen Report zu einem Artikel geschrieben haben, und der PC Chair diskutieren den Artikel. Die PC Members vergleichen ihre Gutachten und erörtern ein gemeinsames Ergebnis.
- **Globale Diskussion:** Nach der lokalen Diskussion existiert meist eine noch zu große Anzahl von Artikeln die würdig wären, bei der Tagung vorgetragen und letztendlich veröffentlicht zu werden. Nun erhalten alle PC Member Zugriff auf alle Reports und Diskussionsforen. Ziel ist es, aus den zur Akzeptanz vorgeschlagenen Artikeln diejenigen zu filtern, die tatsächlich auf der Tagung vorgetragen werden sollen.

Der PC Chair schreibt nach Festlegung der Entscheidung ein Gesamtgutachten für die angenommenen und abgelehnten Artikel (vgl. Schritt 5 “write final report” in Abb. 1.1). Die Autoren erhalten via Email den Hinweis, dass die Gutachten anonym online zugänglich sind. Autoren mit einem akzeptierten Artikel müssen eventuelle Änderungen und Verbesserungsvorschläge einpflegen sowie die endgültige Version des Artikels samt Quelldateien hochladen (vgl. Schritt 6 “submit final version” in Abb. 1.1).

Die akzeptierten Artikel werden anschließend in einem Tagungsband von einem Verlag veröffentlicht und bilden die Grundlage für die Tagung, an der das Komitee, die Vortragenden (Autoren der Artikel) und weitere interessierte Personen teilnehmen. Zu den vorbereitenden Geschäftsprozessen einer Konferenz gehört darüber hinaus die Registrierung als Teilnehmer, die Buchung eines Hotels und eventuelle des Fluges und Beantragung eines Visums. In Kapitel 5 wird näher auf diese Details eingegangen.

1.2.2 Allgemeine Anforderungen

Die allgemeinen Anforderungen an ein Entscheidungssystem hängen von den Einsatzbedingungen und der Komplexität des Systems ab. Sind große heterogene Benutzergruppen zu bedienen, mit häufig wechselnden oder wachsenden Anforderungen der Dienstfunktionalitäten ist ein flexibler Ansatz in Bezug auf Softwaredesign und Softwarearchitektur zu wählen. Ein weiterer Aspekt ist der Bereich der Verwendung. Es muss analysiert werden, für welche Technologien und Softwareplattformen die zu entwickelnde Anwendung zu konzipieren ist. Eventuelle Anwendungsgebiete können sein

- Desktop-Anwendungen,
- internetbasierte Applikationen, oder
- mobile Anwendungen für Handys oder PDAs.

Die Anforderungen an ein Softwaresystem können in die zwei Kategorien der *Benutzersicht* und der *Realisierung und Softwarearchitektur* aufgeteilt werden. Für Entscheidungssysteme, insbesondere für Begutachtungssysteme, lassen sich die nachstehenden allgemeinen Anforderungen aus der Benutzersicht zusammenfassen:

- **Intuitive Bedienbarkeit** der grafischen Benutzeroberfläche (GUI [7]).
- **Globaler Zugriff** auf das System und deren Daten.
- **Kurze Wartezeiten (Latenzzeiten)** von der Anforderung neuer Daten bis zu deren Visualisierung.
- **Transparenz** des Prozesses der getroffenen Entscheidungen.

[7] Graphical User Interface

- **Entscheidungsunterstützung** durch Diskussions-, Bewertungs-, und Auswertungskomponenten.
- **Anpassbarkeit** auf verschiedene Kommunities.
- **Zentralisierte Darstellung** relevanter Daten.
- **Verfügbarkeit** des Dienstes (24x7).
- **Sicherheit** von sensiblen Daten.

Unter Berücksichtigung dieser allgemeinen Anforderungen aus der Sicht eines Dienstbenutzers lassen sich die folgenden allgemeinen Anforderungen an die Realisierung und Softwarearchitektur spezifizieren:

- **Internetbasierte Applikation** zur globalen Bereitstellung von Geschäftsprozessen.
- **Heterogenes verteiltes System** für die Realisierung des Gesamtsystems.
- **Modularität und Wiederverwendbarkeit** von Softwarekomponenten und Diensten.
- **Erweiterbarkeit und Modifizierbarkeit** des Softwaresystems.
- **Plattformunabhängigkeit** zur Installation auf unterschiedlichen Betriebssystemen.
- **Verfügbarkeit** des System.

Die Anforderungen bilden die Basis für die Evaluierung der geeigneten Programmiersprache, des Softwaredesigns, der Softwarearchitektur und der zu verwendenden Software-Bibliotheken.

1.2.3 Online Conference Service (OCS): Anforderungen

Die im Kapitel 1.2.2 dargestellten Anforderungen sind in den gut konzipierten Softwaresystemen von Bedeutung. An den OCS wurden darüber hinaus folgende ergänzende Anforderungen und Leistungsmerkmale gestellt, die zur Entstehungszeit des OCS, im Jahre 1999, maßgeblich für das System waren:

- **Konfigurierbarkeit** zur Laufzeit und die dadurch flexible Anpassung des Dienstes an die Bedürfnisse unterschiedlicher Konferenzen. Dieses beinhaltet die Bereiche GUI, Zugriffsrechte, Personalisierung und Änderung des Geschäftsprozesses durch Ein- bzw. Ausschaltung von Dienstfunktionalitäten.

- **Dokumentenmanagement** für die Verwaltung der eingereichten Artikel und Gutachten. Von zentraler Bedeutung ist die online konfigurierbare Benutzeroberfläche, die es ermöglicht, Formulare für die Einreichung von Gutachten online zu erstellen (siehe Kapitel 4.4). Von großem Interesse ist ebenfalls die Versionierung mittels derer die Historie von der ersten bis zur finalen Artikelversion nachvollzogen werden kann. Das ist für die Gutachter notwendig, um die überarbeiteten Artikelversionen auf die im Gutachten festgelegten Änderungshinweise hin überprüfen zu können.

- **Flexible Erweiterbarkeit** des Entscheidungssystems durch neue Dienstfunktionalitäten oder Teildienste. Integrationen und Modifikationen der Geschäftslogik und der Businessobjekte sollen einfach und übersichtlich durchgeführt werden können.

- **Zugriffskontrollmechanismus** für die Identifizierung und Authentifizierung von Benutzern des Dienstes. Dieses basiert auf einem Benutzer- und Rollensystem (siehe Kapitel 4.2).

- **Personalisierung** der Benutzersicht auf die Dienstfunktionalitäten und die sensiblen Daten. Die Benutzer erhalten in Abhängigkeit von ihren Zugriffsrechten [8] und der Konferenzphase den Status der im Begutachtungsprozess befindlichen Beiträge sowie eine für sie maßgeschneiderte Sicht auf den Dienst.

- **Fristenmanagement** für die Verwaltung und Überwachung von Deadlines zur Einhaltung des geplanten Begutachtungszeitraumes. Sind Fristen für die Erfüllung von Aufgaben abgelaufen, erhalten die Gutachter eine Benachrichtigung. Dieses kann das Verfassen eines Gutachtens sein, ohne dessen die nächste Phase des Begutachtungsprozesses nicht eingeleitet werden kann. Des Weiteren überwacht das Fristenmanagement die einzelnen Phasen der Konferenz, die zu Beginn der Konferenz durch den PC Chair mit Zeitfristen annotiert werden. Dadurch sind automatische Phasenübergänge möglich, bei denen zuvor erlaubte Tätigkeiten, wie z.B. das

[8] Zugriffsrechte erteilen dem Besitzer des Rechtes die Erlaubnis, bestimmte Funktionalitäten zu nutzen.

Einreichen eines Beitrages, den Benutzern automatisch entzogen werden können.

- **Konfliktbehandlung** ermöglicht das Kennzeichnen eines Konfliktes zwischen einem potentiellen Gutachter und den Autoren eines Artikels. Ein Konflikt besteht, wenn der Gutachter dem Autor persönlich nahe steht oder abgeneigt ist, oder selbst Autor des Artikels ist. Die Konflikte werden im gesamten Dienst zur Filterung von den zur Anzeige aufbereiteten Daten berücksichtigt. Somit wird gewährleistet, dass Benutzer mit Konflikten nicht an vertrauliche Daten gelangen.

- Die **Verteilungskomponente** unterstützt den PC Chair während der Delegierung von Begutachtungen. Verteilungen können geplant, von den PC Members selbst vorgeschlagen, oder mit einem Verteilungsalgorithmus unter Verwendung eines Lösungsverfahrens für lineare Probleme automatisch bestimmt werden.

- Die **Benutzerverwaltung** bietet eine Managementkomponente für Benutzerdaten an. Unter anderem können Benutzer angelegt und entfernt werden. Benutzerprofile und Accountdaten sind modifizierbar.

- Die **Kommunikationskomponente** des OCS verfügt über eine Email- und Forenfunktionalität. Automatische Dienstemails werden über vorgeschriebene Emailtemplates realisiert, die zur Laufzeit mit den relevanten Daten gefüllt werden. Der PC Chair ist in der Lage, Mitteilungen an alle Empfänger einer oder mehrerer Rollen zu schreiben. Die Foren dienen als Diskussionsplattform zur Bestimmung der akzeptierten Artikel.

- Die **Entscheidungkomponente** in Form der Evaluationsmatrix bietet eine Übersicht über die Begutachtungen und ermittelt eine Rangliste der Artikel, auf Basis eines gewichteten Mittelwertes. Dieser Mittelwert wird aus den Bewertungen, die von den Gutachtern in ihren Reports bestimmt wurden, berechnet.

- **Verifizierbarkeit** ist mit die wichtigste Anforderung an den OCS. Bei einem System, das ständiger Weiterentwicklung unterworfen ist, ist eine automatische Überprüfung der Funktionalitäten von großem Interesse.

Ende der Neunzigerjahre existierten kaum standardisierte Frameworks für diese Art von Anforderungen. Es war die Zeit, als der Boom um die IT-Branche begann, und proprietäre web-basierte Lösungen ins Leben gerufen wurden und den Markt überschwemmten. Die IT erkannte das Potential, komplexe Anwendungen als Web-Applikationen zu realisieren.

Als Entwicklungsumgebung für das Projekt OCS wurde das Agent Building Center (ABC) [SM99, SMCB96] der Firma METAFrame verwendet. Die modellgetriebene Vorgehensweise des ABC hatte sich bereits im IN [9] Bereich [IT92, ITU93] etabliert und wurde für die Modellierung von Testszenarien für große Telefonanlagen erfolgreich eingesetzt [NMH+01, NSM+01]. Das Konzept des ABC sieht eine vereinfachte, für den Anwendungsexperten verständliche Form der Geschäftsmodellierung vor. Die direkte Einbindung von Anwendungsexperten in die Realisierung war und ist ein wesentlicher Aspekt des ABC [MS04a]. Andere Umgebungen für die Modellierung von Geschäftsprozessen wie *LEU* [DGSZ94, Bra01] oder *MOKASSIN* [HG98, Bra01] richteten sich an Benutzer, die tiefere Kenntnisse in der Programmierung haben mussten. Das *ARIS* [10]-Konzept [Sei02], basierend auf der verbreiteten Modellierungssprache *EPK* [11], beinhaltete die Modellierung von Geschäftsprozessen, aber zum damaligen Zeitpunkt nicht die Generierung der ausführbaren Geschäfztsprozesse. Heutzutage wird dieses durch die Einbindung von UML [12] nur indirekt gelöst. Mittlerweile existieren standardisierte Prozessbeschreibungs- und Ausführungssprachen. Hierzu zählt die im Jahre 2002 von der BPMI [13] veröffentlichte grafischen Notation BPMN [14] [WM08], die seit 2006 offizieller OMG-Standard für die Modellierung von Geschäftsprozessen ist. Die erstellten Modelle lassen sich in Ausführungssprachen wie z.B. der BPML [15] oder BPEL4WS [16] implementieren. XPDL [17] ist eine weitere Prozessbeschreibungssprache, die von der WfMC [18] entwickelt wird.

Mittels des ABC ließen sich schon zur Entstehungszeit des OCS web-basierte Anwendungen strukturiert entwickeln, indem die Geschäftslogik der Anwendung durch gerichtete Graphen modelliert wurde. Änderungen und Erweiterungen an der Logik lassen sich noch heute auf einfache Weise am Kontrollflussgraphen durchführen. Die Realisierung und Wartung von Systemen mittels des ABC stellt ein einheitliches Vorgehen dar [MS09, SN07] und unterstützt Entwicklungs- und Anwendungsexperten gleichermaßen. Des Weiteren lassen sich die Graphen mit temporal logischen Formeln auf ihre Gültigkeit über-

[9] intelligent networks

[10] *Architektur Integrierter Informationssysteme* von Prof. August-Wilhelm Scheer seit 1993 entwickelt.

[11] *Ereignisgesteuerte Prozesskette* von Prof. August-Wilhelm Scheer

[12] Unified Modelling Language

[13] Business Process Management Initiative

[14] Business Process Modeling Notation

[15] Business Process Modeling Language

[16] Business Process Execution Language for Web Services

[17] XML Process Definition Language

[18] Workflow Management Coalition

prüfen. Mit dem Model Checking lässt sich bereits zur Modellierungsphase die geplante Anwendung testen. Mittlerweile wurde das ABC vom JavaABC (jABC) [Nag09] abgelöst. Diese Neuentwicklung des ABCs auf Basis von JAVA [mic09l] entstand am Lehrstuhl für Programmiersysteme der TU Dortmund. Das jABC/ABC wird im Kapitel 3.1 detaillierter beschrieben.

Der OCS ist in der objektorientierten Programmiersprache Java entwickelt und basiert auf der Servlet-Technologie. Die Visualisierung der angeforderten Daten wird überwiegend mittels HTML umgesetzt. OCS baut auf dem Model View Controller (MVC) Konzept auf, das eine Applikation in die drei Bereiche

- Datenmodell,
- Präsentationsebene,
- Programmsteuerung

trennt.

Ziel dieses Konzeptes ist es, Modifikationen und Erweiterungen zu erleichtern. Des Weiteren können Bereiche ausgetauscht oder in anderen Projekten durch ihre Kapselung wiederverwendet werden.

Durch die Verwendung des ABCs wird das MVC Konzept um eine Schicht erweitert. Diese Koordinationsschicht spiegelt die graphische Modellierungsebene des ABC wider. Dieser erweiterte Ansatz des MVC-Modells wird als *Lightweight Process Coordination approach* [MS04b] bezeichnet. Das Konzept *Aggressive model-driven development* (AMDD) [MS04a, MS08] beruht auf dem erweiterten MVC Konzept und ermöglicht Personen ohne tiefere Kenntnisse über die Diensttechnik, Anwendungen zu entwerfen.

Die für die Umsetzung des OCS verwendeten Konzepte, Architekturen und Technologien werden später in dieser Arbeit genauer gezeigt.

1.3 Eigener Beitrag

Die von mir konzipierten und realisierten Entscheidungssysteme bilden die Basis für meine Arbeit. Es wurden dabei grundsätzliche Konzepte und Vorgehenweisen im Bereich der Entwicklung von Begutachtungssystemen entworfen, die im Fokus stehen. Meine Anteile an den im Rahmen meiner Dissertation vorgestellten Ergebnissen umfassen insbesondere die folgenden Bereiche:

- **Enwicklungskonzept**: Die Umsetzung und Beschreibung eines Entwicklungskonzeptes zur Umsetzung von agilen web-basierten Begutachtungssystemen, wie es der OCS repräsentiert, ist ein zentraler Punkt in dieser Arbeit. Der konzipierte Feature-orientierte Ansatz (siehe Kapitel 4.1) fußt auf dem graphbasierten Konzept des ABCs und stellt die Dienstkomponenten in Form von Features in Beziehung. Die dadurch erzielte Benennung und Gruppierung der Dienstfunktionalitäten, und die daraus folgende strukturierte Hierarchie der Features ist der Grundstein für die Personalisierung der Begutachtungssysteme, die mittels eines eigenen Zugriffskontrollmechanismus (siehe Kapitel 4.2) realisiert wurde. Die Repräsentation des Kontrollflusses einer Applikation durch graphbasierte Modelle ermöglicht auf einfache Weise visuell auf neue bzw. geänderte Anforderungen zu reagieren. Diese Vorgehensweise hat sich bereits im Bereich der Entscheidungssysteme bei den Projekten OCS, OJS, LNCS und Transactions etabliert (siehe Kapitel 7) und wurde in [KM05] veröffentlicht.

- **Personalisierung**: Die Konzeption und Realisierung eines Zugriffskontrollmechanismus und dessen Integration in die Entscheidungssysteme ist ein weiterer wesentlicher Punkt. Der Mechanismus stellt einen sicheren Umgang mit sensiblen Daten dar und geht einher mit der graphbasierten Konzeption der Systeme. Eine Erweiterung des Mechanismus durch den Ansatz eines direkten Benutzer/Rechte-Modells bewirkte eine Steigerung der Flexibilität im Bezug auf die Vergabe von Rechten. Der Zugriffskontrollmechanismus und dessen Erweiterung werden im Kapitel 4.2 näher vorgestellt und wurde in [KM06] veröffentlicht.

- **Komponenten-Realisierung**: Des Weiteren war ich verantwortlich für die Konzeption und Realisierung der für die Begutachtungssysteme zugrunde liegenden Features. Basierend auf diese Features kann auf einfache Weise mittels des graphbasierten Ansatzes neue Dienste erstellt bzw. vorhandene erweitert werden. In dieser Arbeit wird auf die Konzepte und Umsetzung zweier Features detaillierter eingegangen. Sie haben einerseits zu einer wesentlichen Arbeitserleichterung der Konferenzleitung und andererseits zur flexiblen Anpassbarkeit an verschiedene Kommunities beigetragen:

  - Der Bidding-Mechanismus, vorgestellt im Kapitel 4.3. Dieses Feature realisiert eine Verteilungskomponente, die auf Basis der linearen Optimierung Lösungen für Verteilungsprobleme bestimmt. Die Komponente hat sich bereits in vielen wissenschaftlichen Konferenzen bewährt, indem sie die Konferenzleitung während der Planung

und Durchführung der Verteilung von Artikeln an die Gutachter unterstützt hat.

- Die XML-basierten, konfigurierbaren Bewertungskomponente, mit deren Hilfe Begutachtungsformulare online erstellt und Auswertungsformeln definiert und integriert werden können (siehe Kapitel 4.4). Diese Flexibilität der Anpassbarkeit von Formularen ist ein wichtiges Kriterium, um Anforderungen der Wissenschaftsgemeinden zu erfüllen.

- **Validierung und Fallstudien**: Der temporal logische Ansatz für die graphbasierte Modellierung des OCS, wurde den bekannten Ansätzen XACML und WS-Policy gegenübergestellt (siehe Kapitel 2.2) und in [KMW08] veröffentlicht. Fallstudien über den Einsatz und die Verwendung des OCS wurden ebenfalls durchgeführt (siehe Kapitel 10) und in [MK02] veröffentlicht. Anhand der Fallstudien lassen sich die realisierten Features auf deren Verwendung hin analysieren und getroffene Designentscheidungen überprüfen bzw. neue bestimmen.

1.4 Aufbau der Dissertation

Im Folgenden wird die Struktur der Dissertation, die in 6 Teile aufgeteilt wurde, erläutert.

Teil I dient zur Einführung in diese Arbeit. Nach dem einleitenden Kapitel 1 wird im Kapitel 2 das technische und wissenschaftliche Umfeld in einer Studie vorgestellt. Die Anwendung OCS und dessen Konzepte werden mit anderen existierenden Systemen verglichen.

Teil II beschreibt die für die Entwicklung des OCS verwendeten Technologien und dient als Basis für die darauf folgenden Teile der Dissertation. In Kapitel 3 werden die für den OCS eingesetzten Frameworks erläutert und die Verifikation von Graphmodellen eingeführt. Das Kapitel 4 stellt die realisierten Konzepte für eine flexible konfigurierbare web-basierte Softwareentwicklung dar, was eine Vorstellung der Verifikation an einem konkreten Beispiel beinhaltet.

Teil III geht auf das Entscheidungssystem OCS im Einzelnen ein. In Kapitel 5 werden Geschäftsprozesse und die wichtigsten Features beschrieben. Kapitel 6 stellt das Design und die Architektur des Dienstes vor. Die Dienstfamilie des OCS und die Analyse verschiedener in der Dienstfamilie verwendeter Technologien werden im Kapitel 7 beschrieben.

Teil IV widmet sich der Darstellung von Erfahrungen im Bereich der Ent-

wicklung einer komplexen Web-Applikation. Design-Entscheidungen und deren Auswirkungen werden in Kapitel 8 aufgezeigt, der bisherige Einsatz des Dienstes in Kapitel 9 dokumentiert, und die Verwendung des OCS durch eine Fallstudie in Kapitel 10 erörtert.

Teil V enthält mit Kapitel 11 eine Zusammenfassung der vorgestellten Arbeit und gibt einen Ausblick auf mögliche Erweiterungen.

Kapitel 2

Verwandte Arbeiten

In diesem Kapitel wird ein Bezug der in dieser Arbeit vorgestellten Themen zu verwandten Arbeiten im wissenschaftlichen und technischen Bereich aufgezeigt. Kapitel 2.1 gibt einen Überblick über andere Begutachtungssysteme (peer review systems) und stellt Unterschiede zum OCS heraus. Der eigene Entwicklungsansatz für Web-Applikationen wird im Hinblick auf die Zugriffssicherheit auf sensible und vertrauenswürdige Daten unter Verwendung von Policies und Model Checking in Kapitel 2.2 mit den Technologien WS-Policy und XACML verglichen. Kapitel 2.3 zeigt Vorgehensmuster (Patterns) für die Durchführung von Begutachtungen auf und setzt diese mit den Vorgehensweisen des OCS in Beziehung.

2.1 Begutachtungssysteme

Seit Ende der neunziger Jahre sind eine Reihe von webbasierten Begutachtungssystemen entwickelt und für internationale Konferenzen eingesetzt worden. Einige von ihnen wurden in einer Zusammenfassung über Conference Management Software verglichen [Sno09], von denen heute nur noch wenige ihren Einsatz finden. Im Bereich der wissenschaftlichen Begutachtungssysteme kommen stetig neue Systeme mit mehr oder weniger großem Erfolg auf den Markt. Im Folgenden wird neben dem Online Conference Service der Fokus auf die in der Wissenschaftsgemeinde bekannten Systeme ConfMaster [Pre09b] [Pre09c], Continue [Kri09], Conference Reviewing System (CRS) [Sof09], CyberChair [Sta01] [vdS09a]/ CyberChairPRO [vdS09b], EasyChair [Vor09], Editor's Assistant (EDAS) [Sch09], Editorial Manager [Ari09], OpenConf [Gro09] und START V2 Conference Manager [GG09] gelegt. In Kapitel 2.1.1 werden die

erwähnten Begutachtungssysteme kurz vorgestellt, um sie anschließend in Kapitel 2.1.2 auf zentrale Features hin zu diskutieren.

2.1.1 Vorstellung verwandter Systeme

Dieses Kapitel dient zur kurzen Einleitung der relevanten Systeme.

ConfMaster

Zwischen 1997 bis 2000 wurde das Begutachtungssystem ConfMan [Pre09a, Pre09b] von Ketil Lund, Pal Halvorsen und Thomas Preuss entwickelt. Im Jahr 2003 wurde ConfMaster auf Basis von ConfMan implementiert und von Professor Thomas Preuss herausgegeben. Die Entwicklung und das Design wurden hauptsächlich von Bastian Böing durchgeführt. ConfMaster [Pre09b] ist eine Web-basierte Applikation, die auf der LAMP [1] Plattform aufbaut. Das Unternehmen Coniant [Pre09c] bietet den Dienst für Konferenzen auf firmeneigenen Servern kommerziell an.

Continue

Die erste Version des Begutachtungssystems wurde von Shriram Krishnamurthi entwickelt und erstmalig im Jahr 2002 für Konferenzen eingesetzt. In der Zwischenzeit ist die neue Version Continue 2.0 [Kri09] im Einsatz, die auf eine Ajax-basierte Architektur beruht. Momentan ist Arjun Guha verantwortlich für die Weiterentwicklung des Systems. Das Begutachtungssystem wird kostenlos angeboten und gehostet.

CRS

Das Conference Reviewing System CRS [Sof09] wird seit 2002 von dem Unternehmen RedWhale Software kommerziell für Konferenzen angeboten. Entwickelt ist die Applikation mit der ASP [2] Technologie und wurde Skript-basiert umgesetzt. Konferenzen werden auf den firmeneigenen Servern von RedWhale verwaltet und gepflegt.

[1] Das Akronym LAMP steht für die Kombination der freie Software Linux, Apache, MySQL und Perl, wobei das P auch für PHP oder Python stehen kann.

[2] ASP steht für Active Server Pages

CyberChair/CyberChairPRO

CyberChair [vdS09a] wurde im Jahr 1996 das erste Mal eingesetzt. Entwickler des Begutachtungsystems ist Richard van de Stadt. CyberChair basiert auf der Veröffentlichung [Nie00] von O. Nierstrasz, in der Pattern für die erfolgreiche Entscheidungsfindung über die Annahme bzw. Ablehnung von Konferenzbeiträgen definiert wurden. Auf die Veröffentlichung wird in Kapitel 2.3 näher eingegangen. CyberChair ist eine freie Software, die von 1996 bis 2000 entwickelt und gepflegt wurde, bis die neue kommerzielle Version CyberChairPRO [vdS09b] fertiggestellt war. Das System ist in der Skriptsprache Python implementiert.

EasyChair

Professor Andrei Voronkov begann im Jahre 2002 mit der Entwicklung von EasyChair [Vor09], einem Konferenzsystem, das mittlerweile eines der meist verwendeten Systeme im wissenschaftlichen Bereich ist. Für die Umsetzung von EasyChair wurde der Entwicklungsansatz LAMP verwendet. Die Benutzung des Begutachtungsystems für Konferenzen ist kostenlos und wird auf EasyChair eigenen Rechnern an der Universität Manchester am Fachbereich Computer Science betrieben. Es handelt sich um ein Skript basiertes System, das die benötigten Funktionalitäten für den Benutzer in einfachster Form bereitstellt.

EDAS

EDAS [Sch09] steht für Editor's Assistent und wurde von Professor Henning Schulzrinne Ende der neunziger Jahre entwickelt. EDAS ist ein kommerzielles System, das auf dem LAMP Modell aufbaut und somit wie EasyChair Skriptbasiert ist. Die Konferenzen werden von zwei kommerziellen Rechenzentren gehosted, eines ist in Dallas, Texas und das andere in San Diego, Kalifornien ansässig. Das System wurde neben Konferenzen und Workshops, ebenfalls für Journals verwendet. EDAS kann auf einen Datenbestand von 100.000 Begutachtern und Autoren zurückgreifen.

Editorial Manager

Editorial Manager [Ari09] ist ein Manuskriptverwaltungssystem für wissenschaftliche Zeitschriften von der Firma Aries (Deutschland), die eine Toch-

tergesellschaft der Aries GmbH aus den USA ist. Editorial Manager wird in mehreren Verlagen verwendet, wie im Springer Verlag Heidelberg. Neben dem OCS, der für die Konferenzreihen zuständig ist, wird bei Springer der Editorial Manager für die Zeitschriftenverwaltung eingesetzt.

OpenConf

OpenConf [Gro09] ist ein Peer-Review Management System, das seit 2002 existiert und von der Zakon Group in den USA vertrieben wird. Es wird eine in den Funktionalitäten eingeschränkte Version kostenlos angeboten. Die kommerzielle Version ist um einige Features reicher und wird über Lizenzen erworben. Zusätzlich zu den Lizenzen kann optional das Hosting der Konferenzen eingekauft werden. Für die Entwicklung der Software kamen die Skriptsprache PHP und die MySQL Datenbank zum Einsatz. OpenConf wurde bereits für zahlreiche internationale Tagungen eingesetzt.

Open Conference/Journal System

Es sei auch das Open Conference Systems [Pro09] erwähnt, eine freie Skript basierte Konferenzmanagement-Software, die von Public Knowledge Project [Sys09] herausgegeben wird. Dieses System dient zur Erstellung des Web-Auftritts einer Konferenz und die Web-Veröffentlichung von wissenschaftlichen Artikeln. Es beinhaltet einen Einreichungsmechanismus und eine Diskussionskomponente für die Nachbesprechung im Anschluss einer Konferenz, allerdings wird der Begutachtungsprozess für Konferenzen nicht unterstützt.

START V2

Rich Gerber und Paolo Gai entwickelten die erste Version START V1 [GG09] zwischen 1998 und 2000. Diese Version wird bis heute kostenlos angeboten und kann auf eigenen Servern installiert werden. Die neue Version START V2 [GG09] entstand ab dem Jahre 2001 und wird seit dem mittels des Unternehmens Softconf.com, dessen Inhaber Rich Gerber und Paolo Gai sind, weiterentwickelt und kommerziell angeboten. Wie bei OpenConf wird auch hier die Nutzung über Lizenzen geregelt und die Software kann entweder auf eigenen Servern installiert oder ein Hosting der Konferenz zusätzlich gebucht werden. START V2 bietet eine Vielzahl von Features an und ist wie der OCS fein-granular konfigurierbar. Beide START Versionen basieren auf der Skriptsprache Perl.

2.1.2 Analyse und Diskussion

Dieses Kapitel geht auf Technische Aspekte und die Features der vorgestellten Dienste ein und stellt einen Vergleich dar.

Der OCS war durch seinen Entwicklungsansatz und den Zugriffskontrollmechanismus bereits im Jahre 2000 das flexibelste Begutachtungssystem in Bezug auf die Konfiguration des Dienstaussehens, des Workflows, der Rollendefinition und der Anpassbarkeit auf verschiedene Wissenschaftsgemeinden. Der OCS gab den Dienstbenutzern eine große Anzahl von Features an die Hand [MK02], die das kooperative Arbeiten immens erleichterten. Die konzipierte und umgesetzte Forum-Komponente auf Basis eines integrierten News-Servers unterstrich zusätzlich die Vorreiterstellung. OCS und OJS setzten Akzente in der Strukturierung und Komfortabilität des Begutachtungsprozesses. Hat OCS maßgeblich den Begutachtungsworkflow für Konferenzen mitbestimmt und andere Dienste in ihrer Feature-Entwicklung beeinflusst, so war der OJS federführend im Bereich der Journals.

Meine Recherche hat ergeben, dass für die meisten Dienste keine Veröffentlichungen existieren, auf die referenziert werden könnte. Neben dem OCS verfügen nur *CyberChair* und *Continue* über Veröffentlichungen. Das Wissen über die einzelnen Dienste wurde auf Basis von Informationen von Internetauftritten, Evaluierung von Demodiensten, eigener Erfahrung durch die Benutzung der Dienste während wissenschaftlicher Konferenzen und durch Diskussionen mit anderen Anwendern erlangt.

Mittlerweile haben sich die Konkurrenzdienste im Bereich der Begutachtungssysteme vom Funktionalitätsumfang her angenähert. Unterschiede lassen sich heutzutage bzgl. der Umsetzung der Systeme, der Flexibilität und der Ausprägung der einzelnen Features bestimmen. Die im vorangegangenen Kapitel kurz vorgestellten Dienste werden mit dem OCS auf Basis des aktuellen Standes hinsichtlich der nachstehenden Kriterien analysiert und diskutiert:

- **Dienstfunktionalitäten** und deren **Flexibilität**.
- **Agilität** und **Erweiterbarkeit** der Systeme.
- **Validierung** der betrachteten Dienste.

Die Kriterien werden einerseits untermauert durch die verwendeten Architekturen, Technologien und Vorgehensweisen, andererseits durch die in Tabelle 2.1 aufgelisteten und für den Vergleich der Dienste betrachteten Features. Die Tabelle 2.2 dargestellte Legende gibt einen Überblick der verwendeten Synonyme.

Features	CM	Co	CRS	CC	EC	EDAS	EM	OC	OCS	SV
Flexible Rollenkonfiguration	+	?	-	-	-	-	+	-	+	-
Bidding	+	+	+	+	+	+	-	+	+	+
Diskussionskomponente	+	+	+	+	+	+	-	+	+	+
Berechnung der Review-Verteilung	-	?	-	+	-	-	-	-	+	-
Tagungsbanderstellung	-	?	-	+	+	-	-	+	+	-
Double blind mode	+	-	+	+	+	+	-	+	+	+
Anonymisierte PC member	-	-	+	-	-	-	-	-	+	-
Konfigurierbare Reportformulare	-	?	+	-	+	-	+	-	+	+
Integrierter Demodienst	-	-	-	-	+	-	-	-	+	-
Feature-orientierter Support	-	-	-	-	-	-	-	-	+	-
Sub-Delegierungen	+	-	+	+	+	+	+	-	+	-
Konfigurierbare E-Mail-Vorlagen	+	-	-	-	+	+	+	-	+	-
Zentraler Dienstzugang	-	?	+	?	+	+	+	-	(+)	-

Tabelle 2.1: Feature-Vergleich der Begutachtungssysteme (Stand Juli 2009)

Synonym	Bedeutung
CM	ConfMaster
Co	Continue
CRS	Conference Reviewing System
CC	CyberChair/CyberChairPRO
EC	EasyChair
EDAS	Editor's Assistent
EM	Editorial Manager
OC	OpenConf
OCS	Online Conference Service
SV	START V2
?	nicht bekannt
-	nicht vorhanden
+	vorhanden
(+)	noch nicht produktiv geschaltet

Tabelle 2.2: Legende zu Tabelle 2.1

Dienstfunktionalitäten und deren Flexibilität

Unterschiede zwischen den Systemen sind in der Anzahl und der Flexibilität bzw. Ausprägungen der Dienstfunktionalitäten der Systeme festzustellen. Tabbelle 2.1 zeigt eine Auswahl an Features auf, die überwiegend die Flexibilität eines Systems repräsentieren und als Basis für den Vergleich der Systeme dienen. OCS ist nach 10 jähriger Existenz eines der flexibelsten Systeme im Bereich der Begutachtungssysteme. Im Folgenden werden die erlangten Kenntnisse aus Tabelle 2.1 genauer diskutiert.

OCS ermöglicht dem Konferenzleiter den Dienst online zu konfigurieren, angefangen bei dem **Look&Feel** der Dienstseiten bis hin zur Erstellung neuer **Begutachtungsformulare** oder zusätzlicher Rollen, die nicht im Pool der vordefinierten Rollen existieren, aber für den Workflow der Wissenschaftsgemeinde essentiell sind. OCS verfügt über ein flexibles Rollen- und Rechte-System mittels dessen auch zur Laufzeit der Konferenz **Rollen neu erzeugt, modifiziert, zugewiesen und Benutzern entzogen werden können**. Wird die erstellte Vergleichstabelle der Begutachtungssysteme betrachtet, dann ist unter dem Feature *Flexible Rollenkonfiguration* zu erkennen, dass neben OCS nur *ConfMaster* und *Editorial Manager* einen Rollenansatz realisiert haben, diese erreichen aber nicht die Flexibilität des im OCS verwendeten Konzeptes. Die Recherche hat ergeben, dass nur der OCS die Möglichkeit zur zusätzlichen Rollendefinition zur Laufzeit realisiert hat und neue Rollen in dem Begutach-

tungsprozess einbinden kann. Zugriffe auf Objekte im Dienst, wie z.B. Artikel oder Reports, sind im Vergleich zu den anderen Diensten feingranularer einstellbar (siehe Kapitel 4.2).

Features wie das **Bidding** [3] und die **Diskussionskomponente** (siehe Tabelle 2.1) sind nur sehr rudimentär in den anderen Systemen vorhanden. Der Editorial Manager verzichtet gänzlich als einziger auf diese Art von Features. Dieses hängt damit zusammen, dass der Dienst für Journals ausgelegt ist. Im Online Journal Service (OJS), einem der OCS-Dienstfamilie angehörigen Journaldienst (siehe Kapitel 7.2), ist dieses ebenfalls nicht im Basis-Workflow enthalten, kann allerdings zugeschaltet werden. Die Möglichkeit der flexiblen Aktivierung von Features hat sich im Rahmen von Konferenzen und Journals in der Vergangenheit bewährt. Die hohe Anzahl von bereits realisierten Features und deren Verwendbarkeit zur Erweiterung eines bestehenden Workflows stellt einen wesentlichen Anteil der Flexibilität für die eigenen Entscheidungssysteme dar. Das Bidding (siehe Kapitel 4.3) wie die Diskussionskomponente sind als sehr flexible Komponenten für die gesamte Dienstfamilie des OCS umgesetzt worden. Das Bidding-Feature des OCS verfügt über eine Verteilungskomponente, die sich von den anderen realisierten Verteilungsmechanismen abgrenzt. Das Feature berücksichtigt viele Randbedingungen für die **Berechnung einer optimalen Verteilung der Begutachtungsaufträge**, wie die vom Gutachter oder Konferenzleiter spezifizierten Konflikte, die Präferenzen, die der Gutachter für jeden einzelnen Artikel gesetzt hat, die Zugriffsrechte auf den entsprechenden Artikel und die bereits vorhandenen Zuteilungen von Artikeln zur Begutachtung. Eine homogene Verteilung kann mittels eines vom OCS verwendeten Algorithmus für die Lösung lineare Gleichungen berechnet werden. Eine Funktionalität für die Berechnung von homogenen Verteilungen der Begutachtungsaufträge wird ebenfalls von CyberChair angeboten (siehe Tabelle 2.1 Feature *Berechnung der Review-Verteilung*).

Die **automatische Erstellung von Konferenzbänden** (proceedings) (siehe Tabelle 2.1 Feature *Tagungsbanderstellung*) wird von CyberChair, EasyChair, und OpenConf unterstützt. Dies bedeutet, dass die Ergebnisse der Konferenz online zu einem Paket zusammengefügt werden. Der OCS geht mit Hilfe seines PPS (Proceeding Production Service) noch einen Schritt weiter. Der PPS erlaubt es online den kompletten Tagungsband zu erstellen, hierzu gehört die Gliederung des Bandes, das Vorwort, diverse Listen der Beteiligten und Indexe. Der PPS generiert die für den Verlag und Druck benötigten Archive entweder aus den LaTeX Sourcen oder aus dem PDF eines jeden Beitrags. Die

[3] Begutachter können ihre Favoriten, die sie gerne begutachten wollen, wählen bzw. Artikel kennzeichnen, die sie nicht bearbeiten wollen, da sie den Autoren beispielsweise gut kennen oder in dem entsprechenden Fachgebiet keine grundlegenden Erfahrungen haben.

Sourcen werden bereits während der Einreichungsphase durch Kompilierung auf ihre Richtigkeit und Vollständigkeit überprüft. Zur besseren Kontrolle des Tagungsbandes wird neben dem Archiv eine PDF-Version des Konferenzbandes erstellt, mittels dessen Fehler direkt ersichtlich sind und dementsprechende Korrekturen vorgenommen werden können.

Das *blind review* [4] und ***double-blind review*** [5] wird in den meisten Diensten unterstützt (siehe Tabelle 2.1 Feature *Double blind mode*). Der **zusätzlich anonyme Begutachtungsprozess** (siehe Tabelle 2.1 Feature *Anonymisierte PC Member*) in Bezug auf die Identität der Gutachter ist allerdings nur bei CRS und OCS berücksichtigt. Die Namen und Email-Adressen der Begutachter werden im OCS anonym behandelt und können im gesamten Begutachtungsprozess nur von der Konferenzleitung erkannt werden. Sind Artikel von Gutachtern eingereicht worden, ist die Anonymität unter den Gutachtern sehr wichtig um die Gleichbehandlung aller Artikel zu gewährleisten.

Konfigurierbare Begutachtungsformulare (siehe Kapitel 4.4) werden in den Diensten CRS, EasyChair, Editorial Manager, START nur rudimentär angeboten (siehe Tabelle 2.1). OCS hebt sich dadurch hervor, dass die gängigen Interaktionselemente, wie z.B. Textfelder, Checkboxen, Fileuploads definiert sind und beliebig zu einem Begutachtungsformular online kombiniert werden können. Somit ist ein eigenes für die jeweilige Konferenz zugeschnittenes Formular online konfigurierbar. Des Weiteren wird die Erstellung einer Formel unter der Verwendung der definierten Elemente und deren hinterlegten Werte angeboten. Mehrere unterschiedliche Formulare für die Gutachten und die Gesamtgutachten der Konferenzleiter sind in einer Konferenz definierbar.

Im Gegensatz zu allen betrachteten Systemen verfügt jede Konferenzinstanz des OCS über einen **eigenen Demodienst**, der das Abbild des Konferenzdienstes darstellt. Sobald der Konferenzleiter den Dienst konfiguriert hat, kann mittels des Demodienstes einerseits die durchgeführte Konfiguration getestet und andererseits die Funktionsweise des originalen Dienstes kennengelernt werden. Das gesamte Komitee erhält Zugriff und kann ein Beispielszenario eines Begutachtungsprozesses durchlaufen. Selbst zur Laufzeit der Konferenz können die Konferenzdaten in den Demodienst eingespielt und eventuelle Tests durchgeführt werden. EasyChair verfügt seit einiger Zeit nun auch über ein solches Feature (siehe Tabelle 2.1 Feature *Integrierter Demodienst*).

Ein weiterer wesentlicher Aspekt für die gute Unterstützung der Wissenschaftsgemeinde ist die Hilfestellung (support) bei auftretenden Fragen. Der OCS ist

[4] Die Autoren erfahren nicht, wer die Gutachten über ihren Artikel geschrieben hat.

[5] Erweitert *blind review* in der Hinsicht, dass die Gutachter die Namen der Autoren nicht kennen.

das einzige Begutachtungssystem, das dem Benutzer mittels einer **Hilfekomponente** online zielgerichtet auf jeder Dienstseite unterstützt. Diese Komponente lehnt sich an den Zugriffskontrollmechanismus an und bereitet die Hilfestellung personalisiert anhand der jeweiligen aktiven Rolle auf. Weiterhin wird eine FAQ angeboten. Darüber hinaus ist der direkte Support durch Personen, die fundiertes Wissen über die Abläufe des OCS aufweisen, gewährleistet. Der Vergleich zu diesem Feature ist ebenfalls in der Tabelle 2.1 Feature *Feature-orientierter Support* dargestellt.

Damit der Begutachtungsprozess in sich transparent wird, ist es notwendig alle Geschäftsprozesse in einem Dienst aufzunehmen, was gewährleistet, dass nichts am Dienst und dem Komitee vorbei erarbeitet wird. **Sub-Delegierungen** stellt ein solches Feature dar, das mittlerweile auch von vielen der anderen Dienste eingesetzt wird (siehe Tabelle 2.1).

Das E-Mail-Feature ist ebenfalls ein wichtiger Punkt im Vergleich der Systeme. **Konfigurierbare E-Mail-Vorlagen** machen den Dienst flexibler in der Anpassung auf neue bzw. verschiedene Konferenzen. Es weisen neben OCS die Anwendungen ConfMaster, EasyChair, EDAS und Editorial Manager dieses Feature auf (siehe Tabelle 2.1).

Der **zentrale Dienstzugang** (Single Point of Entry oder Single Sign on) bietet den Benutzern, die in mehreren Konferenzen involviert sind eine Erleichterung in Hinblick auf die Authentifizierung, um an alle relevanten Dienste zu gelangen. CRS, EasyChair, EDAS, Editorial Manager und OCS besitzen dieses Feature, wobei es für den OCS noch nicht im Produktiveinsatz war(siehe Tabelle 2.1).

Agilität und Weiterentwicklung

Der OCS, ist wie die vorgestellten Systeme, plattformunabhängig. Während OCS in der objektorientierten Programmiersprache Java implementiert ist, sind die übrigen Systeme überwiegend Skript-basiert entwickelt worden. Zum größten Teil basieren die Dienste auf dem LAMP Modell (siehe Kapitel 2.1.1) und sind in den Skriptsprachen Perl, PHP oder Python realisiert. Durch den Feature-orientierten Ansatz und die graphbasierte Modellierung des Kontrollflusses, der auf seine Korrektheit mittels formaler Methoden verifizierbar ist, hebt sich der OCS gegenüber den anderen Systemen in dem betrachteten Kriterium hervor. Softwaresysteme werden ständig Veränderungen unterzogen, die durch neue Anforderungen, geänderte Anwendungskontexte oder neuen Technologien beeinflusst werden. Geschäftsprozesse und Anforderung von Kunden sind in diesem Rahmen die ausschlaggebenden Punkte. Ein System muss

schnell auf geänderte Geschäftsabläufe reagieren können. Diese Agilität ist auf einfachste Weise durch den Einsatz der graphbasierten Modellierung gewährleistet. Der Kontrollfluss ist visuell nachvollziehbar und kann graphisch modifiziert werden. Dadurch wurde konsequent eine MDA und SOA Vorgehensweise praktiziert. Es lassen sich Dienstfunktionalitäten realisieren, die Domänenübergreifend einsetzbar und aus der Sicht der Wirtschaftlichkeit wiederverwendbar sind.

Validierung

Änderungen und Erweiterungen an Softwaresystemen gehören zum Alltagsgeschäft und verursachen hohe Kosten durch Qualitätssicherung. Die Validierung von Arbeitsabläufen auf Graphebene ist ein adäquates Mittel, um diese Kosten zu senken und eine Gewährleistung der Dienstfunktionalitäten nach durchgeführten Modifikationen zu haben, was nicht gänzlich alle Tests überflüssig macht.

Neben dem Model Checking Ansatz beim OCS wird nur noch bei Continue ein Validierungsverfahren eingesetzt, das sich allerdings vom Grundgedanken des OCS-Verfahrens stark unterscheidet. OCS wird modelgetrieben entwickelt. Während der Entwicklungsphase werden die Geschäftsprozesse als Graphen modelliert und auf dieser Basis die Temporal Logischen Formel für das Model Checking parallel spezifiziert. So kann der entstehende Dienst bereits zur Designzeit validiert werden, ohne eine Implementierung vorgenommen zu haben. Die Vorgehensweise stellt einen direkten Bezug zu den Komponenten der Applikation her. Der Ansatz für Continue setzt bei dem fertigen Dienst an [LK04]. Es werden Graphmodelle auf Basis der Dienstsourcen erstellt. Das so entstehende abstrakte Modell wird dann mittels Model Checkings validiert. Hierbei stellt sich die Frage, ob das automatisch generierte Modell adäquat zum zugrunde liegenden Dienst ist und eine sichere Aussage über die Richtigkeit getroffen werden kann.

Dahingegen existieren keine Modellierungstools für die Skript basierten Systeme, mittels derer ohne detailliertere Programmierkenntnisse Änderungen bzw. Erweiterungen sicher und validierbar durchzuführen sind.

Fazit

Während der Gegenüberstellung des OCS mit den betrachteten Systemen hat sich herausgestellt, dass der OCS nach wie vor das flexibelste System ist (siehe Tabelle 2.1). Die langjährige Erfahrung und die Umsetzung der Bedürfnis-

se von wissenschaftlichen Gemeinschaften, die den Dienst verwenden, haben einen großen Teil dazu beigetragen. Sind Funktionalitäten, die im OCS existieren, auch in anderen Diensten vorhanden, wie z.b. der Biddingmechanismus oder die konfigurierbaren Begutachtungsformulare, so konnte festgestellt werden, dass diese für den OCS konsequenter in ihrer Funktionalität und dadurch flexibler konzipiert und realisiert wurden.

EasyChair ist zum heutigen Stand eines der meist verwendeten Begutachtungssysteme im wissenschaftlichen Bereich. Dieses ist mitunter darauf zurückzuführen, dass EasyChair von Beginn an kostenlos angeboten wurde und einen für den Benutzer einfachen Workflow anbietet. Wollten die Benutzer zur Entstehungszeit der Web-Applikationen alles selbst konfigurieren, steht heutzutage die Einfachheit und der geringe Konfigurationsaufwand im Vordergrund.

OCS hat sich aufgrund der verwendeten Konzepte und Funktionalitäten bei Springer als *das* Begutachtungssystem durchgesetzt. Für Konferenzen mit einem Publikationsvorhaben in der führenden Publikationsreihe der Informatik LNCS wird die Benutzung kostenlos angeboten [Hei09b]. Der OCS und die Dienstfamilie werden weiterhin erfolgreich eingesetzt. Durch die konsequente Einhaltung des Feature-orientierten Ansatzes und regelmäßiger Berücksichtigung von Benutzerrückmeldungen (feedback) wuchs der OCS zu einem vielseitigen Begutachtungssysten heran.

2.2 Sicherheit von Web-Applikationen

Der Schutz von sensiblen und vertrauenswürdigen Daten und Ressourcen nimmt eine zentrale Stellung im Bereich der kooperativen Softwaresysteme ein. Entscheidungen werden über bzw. mit zu schützenden Dokumenten getroffen, die über web-basierte Softwaresysteme zur gemeinsamen Arbeit verfügbar sind.

Web-basierte Systeme bearbeiten hunderte von parallelen Anfragen von verschiedenen Nutzern oder sammeln Daten von verteilten Diensten, wodurch die Anforderungen an die Zugriffskontrollen immer komplexer werden. Eine hohe Leistungsfähigkeit und Ausfallsicherheit von Kontrollmechanismen wird durch das Definieren von Policies unterschiedlichster Ausdrucksstärke erzielt. Policies sind Regeln, die Anforderungen und das Verhalten an ein System beschreiben. Eine Policy wäre zum Beispiel: “Nur authentifizierte Benutzer dürfen den internen Bereich der Applikation einsehen.” Durch die stetig wachsende Komplexität von Systemen und die daraus resultierende Komplexität der Zugriffsrechte ist es für Administratoren und Dienstentwickler umso wichtiger, auf ein leistungsfähiges und überschaubares Framework zurückgreifen zu

können, das die Erstellung von Policies unterstützt. Eine wichtige Anforderung für ein solches Framework ist die Anpassbarkeit und Kombinierbarkeit von Policies, um gegenwärtigen Web-Diensten, deren Geschäftsprozessen und Architekturen, die stetig im Wandel stehen, gerecht zu werden.

Die Fähigkeit, Policies zu schreiben und zu verwalten, reicht allerdings nicht aus, um eine fundierte und zuverlässige Zugriffskontrolle zu gewährleisten. Viel wichtiger ist es sicherzustellen, dass die Policies mit der wirklichen Implementierung des zu sichernden Dienstes übereinstimmen. Mit anderen Worten, erfüllen sie die Sicherheitsaufgaben im realen System? Dieses gilt im Hinblick auf

- neue Policies,
- existierende Policies, wenn der zu sichernde Dienst geändert wurde,
- modifizierte Policies auf unveränderten Geschäftsprozessen,
- Änderungen der Policies bei gleichzeitiger Modifikation des zu sichernden Dienstes.

Letzteres stellt die größte Herausforderung an die Gewährleistung der Funktionalität des Zugriffskontrollmechanismus für komplexe Anwendungen dar. Eine Methodik, basierend auf formalen Methoden, die den Zusammenhang und die Übereinstimmung zwischen dem Modell des Dienstes und der Implementierung des Dienstes herstellt, ist somit unerlässlich.

Im Folgenden werden die bekannten Policy Beschreibungssprachen *XACML* und die *Web Services Policy* (WS-Policy) kurz vorgestellt und mit dem auf Basis des jABC konzipierten Ansatzes für das im OCS verwendete Zugriffskontrollkonzept verglichen.

2.2.1 WS-Policy

Das Web Services Policy Framework ist eine Spezifikation des *W3C* [6] [BNB+06, WCL+05]. Es beschreibt ein Rahmenwerk zur Erstellung von Policies und Policy Assertions, das die Eigenschaften, Fähigkeiten und Anforderungen von Web Services beschreibt. WS Policy ist eine Syntax mittels derer Policy Assertions als disjunktive und konjunktive Verknüpfungen zu einer Policy zusammengesetzt werden. Dabei kann es sich um Eigenschaften der Kommunikationsfähigkeiten eines Web Services handeln, oder die Anforderungen an einen

[6] World Wide Web Consortium

anderen Web Service, die dieser für eine Zusammenarbeit zu erfüllen hat. Anwendungsbereiche von Web Services werden Domains genannt. Das Ziel von WS Policy ist die Beschreibung von allgemeinen und unabhängigen Policies, die mittels des WS Policy Framework an domainspezifische Anwendungsbereiche gekoppelt werden. Hierzu werden verschiedene Erweiterungen des Frameworks verwendet, z.B. WS-SecurityPolicy [GDLHBH05] für die Richtlinien zur sicheren Kommunikation, WS-PolicyAssertions [MHBK03] für die Anforderungen an die Textverschlüsselung, etc. oder WS-MetadataExchange [CBBa06] um Metadaten eines Web Service zu spezifizieren. Ein Beispiel können zwei Web Services sein, die bestimmen müssen, welcher Verschlüsselungsmechanismus für ihre Kommunikation verwendet werden soll. Einer von ihnen könnte eine Verschlüsselung mit geringer Sicherheitsstufe fordern und unverschlüsselte Anfragen ablehnen, wobei der andere Service nur eine begrenzte Auswahl an Verschlüsselungsalgorithmen bietet.

Mit Hilfe von WS Policy können solche Anforderungen und Fähigkeiten beschrieben und verwendet werden, um passende Kommunikationsparameter abzuleiten. Mittels Policy Assertions werden solche Eigenschaften definiert. Mehrere Policy Assertions werden zu Policy Alternativen zusammengefasst, indem die Assertions mit `<wsp:All>` oder `<wsp:ExactlyOne>` Elementen umschlossen werden. `<wsp:All>` definiert, dass alle Assertions erfüllt sein müssen, wobei `<wsp:ExactlyOne>` genau eine gültige Assertion verlangt. Die Policy Alternativen schließen sich gegenseitig aus und werden ausgedrückt in einer disjunktiven Normalform (*Normal Form Policy Expression*). Des weiteren werden Policy Expressions mittels *Policy Attachments* mit *Policy Scopes* assoziiert, die eine Sammlung von *Policy Subjects* darstellen. Subjects sind Einheiten, Ressourcen, die durch die Policy beeinflusst werden.

Abbildung 2.1 (aus [BNB+06]) zeigt eine Policy in gekürzter Form, die WS-Security Policy verwendet.

2.2.2 XACML

Die *eXtensible Access Control Markup Language (XACML)* [Mos05] ist 2003 erschienen. Der OASIS [OAS09] Standard wurde erstellt für die Spezifikation von Policies und Eigenschaften auf Basis von XML. Dieser Standard wird eingesetzt, um Bedingungen (constraints) für den Zugriff auf Ressourcen zu definieren. Die Bedingungen selbst werden in Form von Attributen und Datentypen beschrieben und mittels generischer Funktionen überprüft. Neben der Policy gibt es Policy Sets, die durch kombinatorische Angaben (wie *all*, *at least one*, ...) zu sogenannten Policy Sets kombiniert werden können.

```
01 <wsp:Policy xmlns:sp= "http://schemas.xmlsoap.org/ws/2005/07/
securitypolicy"
02   xmlns:wsp= "http://schemas.xmlsoap.org/ws/2004/09/policy" >
03   <sp:TransportBinding>
04     <wsp:Policy>
05       <sp:AlgorithmSuite>
06         <wsp:Policy>
07           <wsp:ExactlyOne>
08             <sp:Basic256Rsa15 />
09             <sp:TripleDesRsa15 />
10           </wsp:ExactlyOne>
11         </wsp:Policy>
12       </sp:AlgorithmSuite>
13       <sp:TransportToken>
14         <wsp:Policy>
15           <sp:HttpsToken RequireClientCertificate= "true " />
16         </wsp:Policy>
17       </sp:TransportToken>
18       <!-- Details omitted for readability -->
19     </wsp:Policy>
20   </sp:TransportBinding>
21 </wsp:Policy>
```

Abbildung 2.1: Beispiel *WS-Policy* policy

Abbildung 2.2 (aus [Gri04]) zeigt die Komponenten von XACML und deren Zusammenhang. Eine eingehende Anfrage (request) wird von dem *Policy Enforcement Point* (PEP) in eine XACML Anfrage transformiert, die Informationen über den Anfragenden, seine Berechtigungen und die angefragte Ressource zusammenfasst (vgl. Schritt "(1)" in Abb. 2.2). PEP kommuniziert mit dem *Policy Information Point* (PIP), der Nutzerinformationen und Kontextinformationen verwaltet (vgl. Schritt "(2)" in Abb. 2.2).

Die vom PIP kommenden Informationen werden in die XACML Anfrage eingearbeitet und diese anschließend an den *Policy Decision Point* (PDP) weitergeleitet (vgl. Schritt "(3)" in Abb. 2.2). Dieser entscheidet unter Berücksichtigung von Policies von dem *Policy Store* (vgl. Schritt "(4)" in Abb. 2.2) über Annahme oder Ablehnung der Anfrage. Wenn zutreffende Policies vorhanden sind, werden diese überprüft und PEP leitet die Authentifizierungsentscheidung weiter (vgl. Schritt "(5)" und "(6)" in Abb. 2.2).

Das Root-Element eines XACML Dokumentes ist eines der folgenden Elemente: `<Rule>`, `<Policy>` und `<PolicySet>`. Eine Policy besteht aus einer oder mehreren Regeln (rules) und einem Kombinationsalgorithmus für die Ergebnisse der Regeln. Policies wiederum können Bestandteil von Policy Sets sein, die aus mehreren `<Policy>` und `<PolicySet>` Elementen zusammengesetzt sein können. Für die Bestimmung eines eindeutigen Ergebnisses können auch `<PolicySet>` Kombinationsstrategien definiert werden. Kombinationsalgorithmen für Regeln und Policies sind:

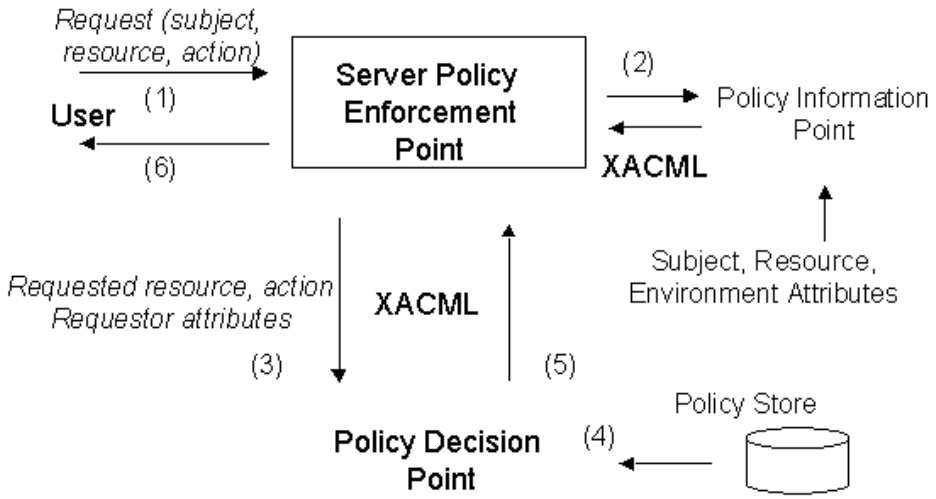


Abbildung 2.2: Verarbeitung einer Anfrage in XACML

- **Deny-overrides** Sobald eine Regel oder Policy `Deny` festsetzt, ist das Gesamtergebnis `Deny`.
- **Permit-overrides** Sobald eine Regel oder Policy `Permit` festsetzt, ist das Gesamtergebnis `Permit`.
- **First applicable** Die erste einsetzbare Regel oder Policy die gefunden wird bestimmt das Gesamtergebnis.
- **Only-one-applicable** Dieser Kombinationsalgorithmus kann das Ergebnis von genau einer Policy eines Policy Sets abhängig machen. Ist genau eine Policy anwendbar, ist der Algorithmus abgeschlossen. Wenn mehrere Policies zutreffen, ist das Ergebnis `Indeterminate`. Sobald keine passende Policy gefunden wird, wird `NotApplicable` gemeldet. Dieser Algorithmus kann nur für Policy Sets verwendet werden und ist nicht geeignet für nur eine Regel.

Die Verwendbarkeit einer Regel oder Policy ist abhängig von dem `<Target>` Element. Dieses spezifiziert, für welche Subjekte, Ressourcen oder Kontexte die Regel oder Policy definiert wurde. Das Target ist das erste Element, das überprüft wird, wenn eine Policy bearbeitet wird. Wenn das Target mit einer Anfrage übereinstimmt, ist die Evaluierung der Policy abgeschlossen, andernfalls wird das Target der nächsten Policy überprüft.

Abbildung 2.3 (aus [Mos05]) zeigt ein Beispiel für eine Policy. Es beinhaltet eine Regel (Zeile 12–31), deren Bedeutung in `<Description>` beschrieben ist

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
03   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04   xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
05   http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd"
06   PolicyId="urn:oasis:names:tc:example:SimplePolicy1"
07   RuleCombiningAlgId="identifier:rule-combining-algorithm:deny-overrides">
08   <Description>
09     Medi Corp access control policy
10   </Description>
11   <Target/>
12   <Rule RuleId= "urn:oasis:names:tc:xacml:2.0:example:Simple Rule1" Effect="Permit">
13     <Description>
14       Any subject with an e-mail name in the med.example.com domain
15       can perform any action on any resource.
16     </Description>
17     <Target>
18       <Subjects>
19         <Subject>
20           <SubjectMatchMatchId="urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match">
21             <AttributeValueDataType="http://www.w3.org/2001/XMLSchema#string">
22                 med.example.com
23             </AttributeValue>
24             <SubjectAttributeDesignator
25              AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
26              DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"/>
27           </SubjectMatch>
28         </Subject>
29       </Subjects>
30     </Target>
31   </Rule>
32 </Policy>
```

Abbildung 2.3: Beispiel einer XACML Policy

(Zeile 13–16). Das leere `<Target>` Element in Zeile 11 kennzeichnet, dass die Policy sich auf jede Ressource bezieht.

2.2.3 Policies im OCS

Bevor die aufgezeigten Ansätze in Kapitel 2.2.4 diskutiert werden, wird zuvor kurz auf die eigene Vorgehensweise eingegangen, die detaillierter in den Kapiteln 3.1, 4.1 und 4.2 beschrieben ist.

Policies im OCS definieren das Zusammenarbeiten von Dienstkomponenten beziehungsweise Diensten und ihren Abhängigkeiten. Die Geschäftsprozesse des OCS werden als gerichtete Graphen modelliert. Abbildung 2.4 zeigt eine vereinfachte Darstellung des für das Anzeigen der Artikelliste verantwortlichen Geschäftsprozesses. Die Knoten (SIBs [7]) des Kontrollflussgraphen (SLG [8])

[7] Service Independent Building Block
[8] Service Logic Graph

sind Funktionalitäten eines Dienstes, die selbst Dienste sein können (siehe Kapitel 3.1.1). Die Funktionalitäten in Form von SIBs sind im Falle des OCS in Java geschrieben und durch den komponentenbasierten Ansatz auch projektübergreifend wiederverwendbar.
Eine einfache Policy für den rollenbasierten Zugriff auf die Artikelliste lautet:

Nur Nutzer die als Konferenzleiter in dem OCS angemeldet sind dürfen alle Artikel lesen.

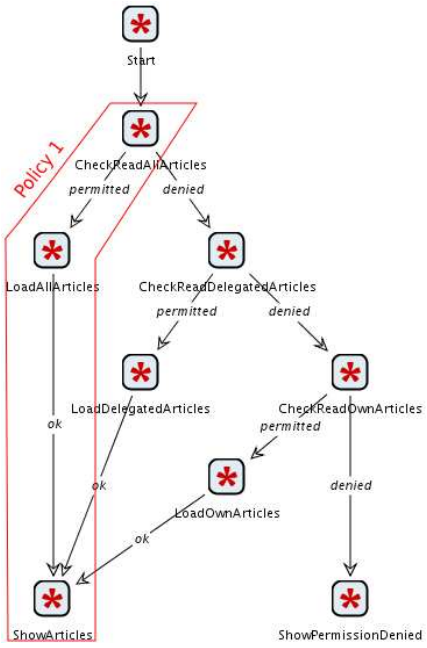


Abbildung 2.4: `ReadArticleList` SLG - in vereinfachter Form

In der Abbildung 2.4 ist der Kontrollfluss für die Anzeige von allen Artikeln durch die Kennzeichung *Policy 1* hervorgehoben. Die Implementierung des SIBs `CheckReadAllArticles` überprüft, ob der Benutzer das erforderliche Recht hat, den SIB `LoadAllArticles` auszuführen.

Mit dem jABC Plugin FormulaBuilder [JMS06] können Policies als logische Bedingungen graphisch modelliert werden, die mittels des GEAR [BMRS08, BMRS06] Model Checkers gegen den Kontrollflussgraphen einer Anwendung geprüft werden. Abbildung 2.5 zeigt *Policy 1* als Formelgraphen.

Der temporale Operator DIAB (*diamond backward*) verhält sich wie ein EX Operator mit umgekehrtem Transitionspfeil. Wir nennen dieses Verhalten EXB,

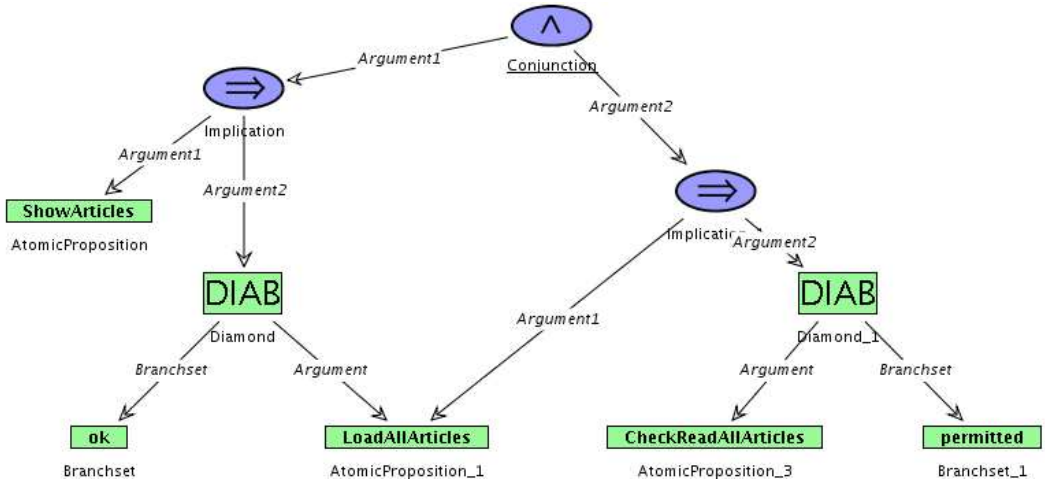


Abbildung 2.5: FormulaBuilder - Graphische Modellierung der Policy 1

das besagt, dass mindestens ein Vorgänger SIB (Knoten) existiert, mit dem Namen, der rückwärts über einen branch (Kante) mit der Benennung erreichbar ist.

Der in Abbildung 2.5 gezeigte Formelgraph wird in CTL [CGP01] Syntax übersetzt:

```
(('ShowArticles => EXB[ok] 'LoadAllArticles) ∧
('LoadAllArticles => EXB[permitted] 'CheckReadAllArticles))
```

Diese Bedingung wird anschliessend intern für den GEAR Model Checker in den modalen μ-Kalkühl [CGP01] übersetzt.

Abbildung 2.6 zeigt, dass der Graph die Bedingung erfüllt. Alle SIBs, die **Policy 1** erfüllen, sind grün und mit einem Kasten markiert worden.

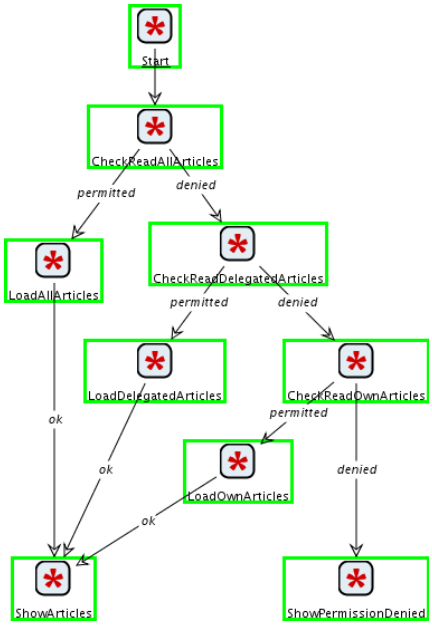


Abbildung 2.6: `ReadArticleList` Feature ist Policy 1 konform

Abbildung 2.7 wiederum zeigt einen Graphen, bei dem die `ok` Kante zwischen `LoadAllArticles` und `ShowArtiles` auf `LoadDelegatedArticles` umgebogen wurde. Der Graph ist mit der `Policy 1` nicht länger konform und die SIBs, die die Policy verletzen, werden gekennzeichnet.

2.2.4 Vergleich der Ansätze

XACML und WS Policy sind beides Sprachen zur Beschreibung von Policies, die es erlauben, aus einfachen, komplexe Policies zusammenzusetzen. Mittels XACML werden Autorisierungsentscheidungen getroffen und die Zugriffe auf Dienste und Ressourcen gesteuert. WS-Policy hingegen verwendet einen allgemeineren Ansatz und benötigt separate domänenspezifische Erweiterungsmodule zur Erstellung von Policies. XACML Policies arbeiten auf Attribute von Anfragen und sind direkt von der Semantik der verwendeten Datentypen abhängig. Dem gegenüber arbeitet WS-Policy mit spezialisierten Bedingungen (Assertions), die in verschiedenen Spezifikationen definiert sind. Dieses hat zur Folge, dass es für Anwendungen, welche WS-Policy einsetzen, schwierig zu entscheiden ist, ob zwei Policies vergleichbar, gleichbedeutend oder widersprüchlich sind, oder ob eine Behauptung eine andere inkludiert.

XACML, wie auch WS-Policy berücksichtigen nur lokale Policies, die Bedin-

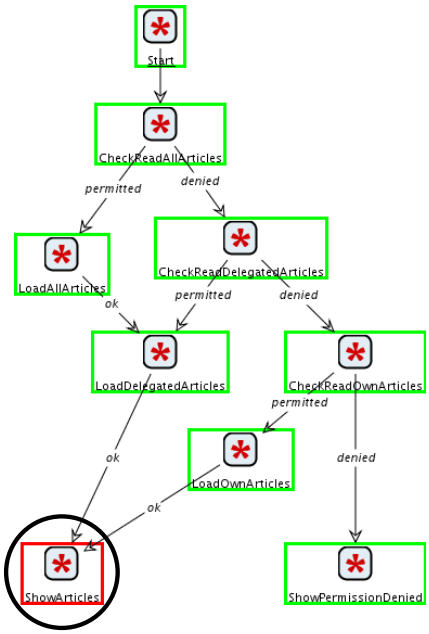


Abbildung 2.7: Verstoß der Policy 1 im geänderten `ReadArticleList` Feature

gungen für eine bestimmte Stelle in einem Prozess definieren. In Bezug auf Prozesse, wie es für Web Anwendungen der Fall ist, ist der Aspekt der globalen Policies von großer Bedeutung. Sie ermöglichen Teile eines Gesamtprozesses in Beziehung zu setzen, um deren Eigenschaften und Abhängigkeiten zu definieren.

Ein anderer sehr wichtiger Punkt für den erfolgreichen Einsatz von Policies ist der Zusammenhang zwischen den geschriebenen Policies und der eigentlichen Implementierung der Anwendung, für die sie definiert wurden. WS-Policy sagt nicht aus, wie Policies auf Dienste angewendet werden sollen. Der XACML Ansatz definiert hingegen den *Policy Enforcement Point* 2.2, der mittels Policies Anfragen auf Dienste oder Ressourcen überprüft. Dieses wird im OCS mit einem eigenen Zugriffskontrollmechanismus umgesetzt, der in den Kapiteln 2.2.3 und 4.2 näher beschrieben ist.

Die Möglichkeit, Policies zu definieren und diese auf einen Dienst anwenden zu können, ist allerdings nur ein Schritt, die Sicherheit sensibler Daten zu gewährleisten. Der zentralere Aspekt für die Sicherstellung der Funktionalität eines adäquaten Zugriffskontrollmechanismus, wie er für den OCS konzipiert wurde, ist die Validierung der Policies. Die Validierung geschieht im direkten Bezug auf die Geschäftsprozesse einer Anwendung, die graphisch die Komponenten der Implementierungen darstellen. Somit können Policies auf der Implemen-

tierung einer Applikation verifiziert werden. Weder WS-Policy, noch der originale Ansatz des XACML zeigen eine Möglichkeit der Validierung der Policy und die daraus folgende Garantie auf Gültigkeit der definierten Policies. Aus jahrelanger Erfahrung im Bereich der Softwareentwicklung gesehen, ist es ein schwieriges Unterfangen, Policies ohne geeignete automatisierte Validierungsmechanismen zu prüfen. Die Komplexität der Policies steigt mit der Zunahme an Dienstfunktionalitäten. Software-Applikationen sind von Natur aus einem stetigen Wandel unterzogen, der notwendige Prüfungen der Konformität der Policies mit sich führt. Es ist sehr schwierig, große Regelsätze manuell zu verwalten und diese gültig zu halten. Insbesondere die Bearbeitung von großen XML Dateien, wie die XML-basierten Ansätze XACML und WS-Policy aufzeigen, bilden für Menschen eine große Herausforderung. Sicherlich können XML Dateien auf ihre Wohlgeformtheit hin gegen ein XML-Schema oder eine XML-DTD [9] geprüft werden. Dieses hat aber keinen Aussagecharakter in Bezug auf die Gültigkeit der Policies auf der zu sichernden Applikation.

XACML und WS-Policy haben umfassende Spezifikationen und Richtlinien für das Definieren und Verwenden von Policies. Allerdings benötigen Ansätze wie in [HB08] komplexe Transformationen und Verschlüsselungen für die Policies, um diese mittels Model Checking validieren zu können.

Es ist entscheidend wichtig, eine integrierte Lösung für die Entwicklung von Applikationen zu besitzen die es ermöglicht, von Beginn einer Entwicklung an kontinuierlich Policies zu überprüfen und im Verlauf der Entwicklung anzupassen, ohne die Gültigkeit der Policies zu verlieren. Diese Vorgehensweise hat sich in der Umsetzung, Weiterentwicklung und Pflege des Entscheidungssystems OCS und seiner Dienstfamilie bewährt. Insbesondere kurzfristig aufgetretene Änderungswünsche während bereits laufender Konferenzen konnten mit dem vorgestellten Konzept schnell umgesetzt, validiert und für die Konferenznutzer sicher zur Verfügung gestellt werden.

2.3 Patterns für Begutachtungsprozesse

Aufgrund der großen Anzahl von zu begutachtenden Artikeln und den gegenüberstehenden kleinen Programmkomitees sind effiziente Begutachtungsprozesse von großem Interesse. Vorteile durch die Optimierung von Begutachtungsprozessen sind:

[9] Die Dokumenttypdefinition (Document Type Definition), wie das XML-Schema, legen die Struktur eines Dokuments fest.

- Kostenminimierung durch Wegfall von unnötigem Zeitaufwand durch nicht organisierte Diskussionstreffen.
- Die begrenzte Zeit der Gutachter, die in der Regel einer wissenschaftlichen Kommunitie angehören und in diesem Rahmen ehrenamtlich Gutachten verfassen.
- Die homogene Verteilung der Begutachtungsaufgaben an die Gutachter.
- Die Auswahl der wirklich besten Artikel aus der Menge der Einreichungen, dieses impliziert die Begutachtung von Artikeln durch Experten in dem jeweiligen Fachgebiet.
- Gleichbehandlung aller Beiträge, was eine strikte Einhaltung des Begutachtungsprozesses für jeden einzelnen Beitrag voraussetzt. Dies gilt auch für Beiträge von Komiteemitgliedern.

Bereits Ende der Neunzigerjahre wurden Erfahrungen aus durchgeführten Begutachtungsprozessen für Konferenzen gesammelt und Konzepte für eine effektive Vorgehensweise niedergeschrieben. In dem bekannten Artikel *Identify the Champion* [Nie00] sind mehrere Pattern spezifiziert, die eine Optimierung und eine Verbesserung der Diskussion über Annahme oder Ablehnung von Artikeln darstellen, die in verschiedenen Konferenzen eingesetzt wurden. Das Begutachtungssystem CyberChair [Sta01] [vdS09a] basiert gänzlich auf die in [Nie00] beschriebenen Pattern. Um derartige Basis-Pattern im OCS abzudecken, hat es sich als günstig erwiesen, flexible Features zu entwickeln. Die Pattern werden im nachfolgenden Kapitel 2.3.1 vorgestellt, um sie anschließend in Kapitel 2.3.2 im Hinblick auf den OCS zu analysieren und die Erkenntnisse in Kapitel 2.3.3 aufzuzeigen.

2.3.1 Vorstellung der Pattern

Das grundlegende Ziel ist es, Gutachter so früh wie möglich im Begutachtungsprozess als Befürworter ("champion") von Artikeln zu manifestieren, die sich für die Annahme des Artikels während des Programmkomiteetreffens einsetzen würden. Diese Treffen dienen zur Bestimmung der Artikel, die zur Konferenz zugelassen werden. Im Folgenden werden diese Pattern in gekürzter Form beschrieben, um sie mit den heutigen Erfahrungen, die ich mit dem OCS und dessen Dienstfamilie gesammelt habe, zu diskutieren.

Pattern: *Identify the Champion*
Dieses Pattern behandelt das grundsätzliche Problem einer Konferenz mit einer großen Anzahl von Einreichungen und der begrenzten Zeit für die Entscheidungsfindung. Wie sollte der Begutachtungsprozess entworfen sein, damit während des Programmkomiteetreffens die besten Artikel akzeptiert werden? Als Lösung wird jeder Gutachter dazu aufgefordert sich festzulegen, ob er für einen Artikel während des Programmkomiteetreffens ein Befürworter ist, oder nicht. Das Ziel ist es, die Diskussion über die Annahme oder Ablehnung auf Artikel mit Befürwortern zu fokussieren, da davon ausgegangen wird, dass diese Artikel eine größere Wahrscheinlichkeit haben angenommen zu werden, als diejenigen ohne Verteidiger. Dieses Pattern ist Voraussetzung für die nachstehenden Patterns.

Pattern: *Experts Review Papers*
Um die besten Artikel selektieren zu können ist es wichtig, eine Strategie für die Verteilung der Artikel an das Programmkomitee (engl. program committee) zu verfolgen. Eine gute Beurteilung und Selektion kann nur durch einen Experten des jeweiligen Fachgebietes durchgeführt werden. Ein einfacher Weg ist es, die Fachgebiete und Kompetenzen der Gutachter als Themengebiete für die Einreichungen festzulegen und die Autoren im *Call for Paper* [10] (CFP) dazu aufzufordern, ihre Artikel diesen Fachgebieten zuzuordnen.

Pattern: *Champions Review Papers*
Dieses Pattern ist konkurrierend zu *Experts Review Papers* in dem Sinne, dass auch hier kompetente Gutachter zu bestimmen sind. Das Ziel ist es, die Chance eines jeden Artikels zu maximieren, einen Befürworter zu erhalten. Die Gutachter werden dazu aufgefordert, selbst Artikel für die Begutachtung auszuwählen. Ein Weg ist es, im CFP die Autoren zu einer Absichtserklärung für eine Artikeleinreichung aufzufordern, indem sie mindestens eine Woche vor der eigentlichen Einreichungsfrist des Artikels den Titel, Autoren, Kontaktadressen und Schlagwörter schicken. Mittels dieser Informationen können dann die Gutachter die gewünschten Artikel wählen. Womit die Zuteilungen bereits feststehen, sobald die Einreichungsfrist der Artikel abgelaufen ist und ohne Zeitverlust mit der Begutachtung der Artikel begonnen werden kann.

Pattern: *Make Champions Explicit*
Dieses Pattern besagt, dass bereits vor dem Programmkomiteetreffen dieje-

[10] Es handelt sich um ein Beitragsaufruf, mit dem zur Einreichung einer wissenschaftlichen Arbeit zur Veröffentlichung aufgefordert wird.

nigen Gutachter bestimmt werden, die Befürworter von Artikeln sein werden. Eine zur Auswahl vorgegebene Bewertung wie “strong accepted” oder “good paper” können unterschiedlich interpretiert werden. Eine hohe Benotung sagt nicht zwangsläufig aus, dass der Gutachter auch Befürworter des Artikels ist und diesen in der Diskussion verteidigen wird. Die vordefinierten Begutachtungsformulare sollen explizit den jeweiligen Gutachter fragen, ob dieser Befürworter des Artikels ist oder nicht.

Pattern: *Identify the Conflicts*
Es wird davon ausgegangen, dass alle oder fast alle Gutachten eingegangen sind und Vorbereitungen für ein reibungsloses Programmkomiteetreffen durchgeführt werden müssen. Es ist wichtig, im Vorfeld des Treffens die umstrittenen Artikel zu eruieren. Das Ziel einer Rangliste und Klassifizierung der Artikel ist es, das Komiteetreffen zu strukturieren, indem Artikel in Gruppen mit dem selben Diskussionsgrund zusammengefasst werden. Eine einfache und effektive Vorgehensweise ist die Gruppierung der Artikel entsprechend ihrer höchsten und niedrigsten Punktzahl mittels einer Kennung aus zwei Buchstaben A-D, wodurch sich die Gruppen AA, AB, AC, AD, BB, BC, BD, CC, CD, DD ergeben. Diese lassen sich durch die Vereinigung von AA mit AB, CC mit CD und mit DD zu sieben Gruppen zusammenlegen, was zu einer Vereinfachung der Diskussion führt.

Pattern: *Identify Missing Champions*
Da PC-Treffen sehr aufwendig und kostenintensiv, nicht wiederholbar und die Entscheidungsprozesse sehr stark von der Identifikation der Befürworter abhängig sind, ist es von sehr großem Interesse, potentielle Probleme im Vorfeld festzustellen. Gutachten können zu spät oder erst kurzfristig geschrieben sein. Befürworter können verspätet zur Diskussion kommen, oder nicht erscheinen. Die Teilnahme und Vorbereitung aller Befürworter und die Einreichung aller Gutachten muss frühzeitig abgesichert werden. Potentielle Befürworter, die nicht an dem PC-Treffen teilnehmen können, müssen im Vorfeld durch andere bzw. zusätzliche Gutachter ersetzt werden, das kann auch unter Umständen die Erstellung von neuen Gutachten nach sich ziehen.

Pattern: *Champions Speak First*
Nachdem die Pattern *Make Champions Explicit*, *Identify the Conflicts*, *Identify Missing Champions* eingesetzt wurden, ist es nun an der Zeit, das Programmkomiteetreffen durchzuführen. Es ist wichtig, schnellstmöglich die Befürworter der Artikel zu identifizieren, da ansonsten sehr viel Zeit für Diskussionen über

Artikel verschwendet wird, die keine Chance haben, akzeptiert zu werden. Einige grundlegende Regeln für die Diskussion sind zu bestimmen, um einen fokussierten Ablauf zu gewährleisten. Die Artikel sollen in Gruppen diskutiert werden, entsprechend des Pattern *Identify the Conflicts*. Da das Treffen in erster Linie dem Akzeptieren von Artikeln dienen soll und nicht der Ablehnung, ist es wichtig, von einem Befürworter des Artikels eine kurze Zusammenfassung zu erhalten, bevor Gegner des Artikels sprechen. Sollte kein Befürworter existieren, wird geraten, festzustellen warum niemand vorhanden ist. Falls kein Befürworter ausgemacht wird, kann der Artikel schnell abgelehnt werden. Ebenfalls wird geraten jedem Teilnehmer des Treffens Kopien von allen Gutachten zu geben, unter Berücksichtigung von Konflikten. Somit können Gutachter die Diskussionen über Artikel besser verfolgen in denen sie nicht direkt involviert sind.

Pattern: *Consensus on PC Papers*
Der Fokus dieses Patterns liegt auf der Vorgehensweise für die Bewertung von Artikeln die von Programmmitgliedern geschrieben wurden. Artikel von Porgrammmitgliedern dürfen nicht bevorzugt behandelt werden. Eine Konferenz, die den Anschein erweckt, dass Artikel von Programmmitgliedern leichter akzeptiert werden als die von anderen Autoren, wird nicht als seriös angesehen. Ein Artikel eines Komiteemitgliedes muss für seine Akzeptanz mindestens einen Verfechter haben und darf von keinem Experten abgelehnt werden. Die Entscheidungsfindung ist unter Ausschluß des Komiteemitgliedes durchzuführen.

2.3.2 Vergleich mit OCS

Der OCS ist entwickelt worden, um alle Beteiligten an einer Konferenz zu unterstützen. Angefangen bei dem Einreichungsmechanismus für die Autoren bis hin zur Erstellung des Tagungsbandes durch den Leiter der Konferenz. Der OCS spiegelt den gesamten Begutachtungsprozess von wissenschaftlichen Arbeiten wider und dient zur Vorbereitung der eigentlichen “physikalisch” stattfindenden Konferenz. Um den Anforderungen an einer effizienten und optimalen Unterstützung zu gewährleisten, wurden im OCS Features entwickelt, die den Begutachtungsprozess durch vordefinierte Prozesse abbilden und teilweise automatisieren.

Der OCS ist ein online konfigurierbares, Web-basiertes Begutachtungssystem, das in Bezug auf das Design der Seiten, der Workflows und der Inhalte von Formularen für verschiedenste Konferenzanforderungen anpassbar ist. Durch

die konfigurierbaren Begutachtungsformulare, welche in Kapitel 4.4 beschrieben werden, lassen sich Elemente definieren und mit Werten hinterlegen, die für die Berechnung einer Note verwendet werden können. Die für die Berechnung benötigte Formel ist ebenfalls frei definierbar. Konferenzen, die nicht das vordefinierte Gutachtenformular verwenden wollen, können so ihr individuelles Formular erstellen. Mit dieser Funktionalität können die Pattern *Identify the Champion* und *Make Champions Explicit* durch direktes Fragen, ob der Gutachter den Artikel während des Komiteetreffens verteidigen würde, erfüllt werden. Eine weitere Möglichkeit ist die Befragung jedes Gutachters in den Foren. Jeder Artikel verfügt über ein eigenes Forum, das in erster Linie von den Gutachtern des jeweiligen Artikels zur Diskussion verwendet wird.

Im OCS können beliebige Artikelkategorien spezifiziert werden, um die Artikel in Gruppen zu gliedern. Diese sind in der Regel Themengebiete von Experten, die Mitglieder in dem Programmkomitee der Konferenz sind. Durch die von den Autoren während der Einreichung durchgeführten Einstufung der Artikel in die Kategorien, sind die Artikel an Experten zur Begutachtung zuweisbar. Dieses Vorgehen entspricht dem Pattern *Experts Review Papers*.

Das Pattern *Champions Review Papers* ist im OCS durch die Features "Abstract Submission" und "Bidding" realisiert. Autoren reichen mindestens eine Woche vor der eigentlichen Artikeleinreichung eine Zusammenfassung ihres Artikels ein und bekunden ihr Vorhaben, einen Artikel zur Konferenz einzureichen. Mit dem "Bidding" Feature bieten die Gutachter anschließend unter Verwendung der Zusammenfassungen und weiteren Artikeldaten auf die Artikel. In dem Entscheidungssystem Online Journal Service (OJS), der der Dienstfamilie des OCS angehört, besteht die weitere Möglichkeit mit dem Feature "Nomination" Herausgeber (Editoren) dazu aufzufordern, eine Liste von Gutachtern im Dienst zu spezifizieren. Dies ist hilfreich, da sich die Editoren in den Fachgebieten auskennen und Kontakt zu entsprechenden Experten haben.

Das Pattern *Identify the Conflicts* unterscheidet sich von dem des OCS. Das OCS Feature "Evaluation Matrix" zeigt die Artikel in einer Rangliste an, die durch den gewichteten Mittelwert der Noten der Gutachten bestimmt wird. Zusätzlich wird eine Varianz der Gutachten bezüglich eines Artikels berechnet, die die Übereinstimmung der Gutachternoten aufzeigt. In Abhängigkeit der Platzierung des Artikels und dessen Varianz können die Artikel in die Gruppen "discussion", "accepted", "proposed accepted", "proposed rejected" oder "rejected" eingeordnet werden. Entscheidungen über Artikel mit einer geringen Varianz, was ein Konsens der Gutachter bedeutet, können schnell getroffen werden. Eine hohe Varianz zeigt den Bedarf einer Diskussion auf, unabhängig davon ob es sich um einen Artikel mit hoher oder niedriger Platzierung in der

Rangliste handelt. Dadurch können auch schlechter eingestufte Artikel noch eine Chance zur Annahme erhalten, was bei der alleinigen Berücksichtigung der Rangliste nicht der Fall wäre.

Programmkomiteetreffen von Konferenzen werden heutzutage überwiegend mittels Kommunikationsmedien wie Foren, Email oder Chat durchgeführt. Die Problematik des Patterns *Identify Missing Champions* ist nicht mehr so relevant, wie sie es noch vor einigen Jahren war. Der OCS stellt hierfür eine Forumkomponente zur Verfügung mit dessen Hilfe Komiteetreffen entweder gänzlich online stattfinden oder zumindest vorbereitet werden können. Komiteemitgliedern kann gezielt Zugriff auf die Foren gegeben werden. Das Kommunitie ATPN [11], diskutiert z.B. die Artikel mittels der Forumkomponente und bereitet so das eigentliche Programmkomiteetreffen vor. Diese Vorgehensweise hat den Vorteil, dass die Teilnahme aller Gutachter eines Artikels am Treffen nicht zwingend notwendig ist.

Im OCS diskutieren erst die Gutachter eines Artikels über dessen Annahme oder Ablehnung, wie es im Pattern *Champions Speak First* beschrieben ist. Die Reihenfolge wer die Diskussion anfängt, z.B. die Befürworter des Artikels, kann vom Konferenzleiter im Forum angefordert werden. Falls es zu keinem Konsens kommt, kann die Diskussion auf zusätzliche oder alle Mitglieder erweitert werden.

Ein generelles Problem stellen die Artikel von Programmkomiteemitgliedern dar, die im Pattern *Consensus on PC Papers* behandelt werden. Der OCS bietet eigens für diese Art von Problem neben den Prozessen des *blind review* und *double-blind review*, zusätzlich den anonymen Begutachtungsprozess in Bezug auf die Identität der Gutachter an. Diese werden im OCS anonym behandelt und erfahren weder die Verfasser der anderen Beurteilungen, noch die Teilnehmer an einer Diskussion.

2.3.3 Fazit

Die in den Pattern *Identify the Champion*, *Experts Review Papers*, *Champions Review Papers*, *Make Champions Explicit* und *Champions Speak First* gezeigten Vorgehensweisen sind im OCS vorhanden und anwendbar, wie in Kapitel 2.3.2 beschrieben.

Der OCS setzt die Pattern *Identify the Conflicts*, *Identify Missing Champions* und *Consensus on PC Papers* dahingegen anders um, wie bereits in Kapitel 2.3.2 aufgezeigt. Durch die Berechnung der Varianz, welche die Überein-

[11] ATPN steht für Application and Theory of Petri Nets

stimmung der Gutachter in Bezug auf deren Benotung darstellt, ist die Feststellung von Konflikten im OCS effizienter als der Lösungsvorschlag im Pattern *Identify the Conflicts*. Konflikte können direkt erkannt werden. Durch die Verwendung der Diskussionsforen des OCS kann entweder das Programmkomiteetreffen "virtuell" stattfinden oder das "physikalische" Treffen vorbereitet werden. Die Foren stellen für das Pattern *Identify Missing Champions* eine kostensparende und effiziente Lösung dar. Ein PC-Mitglied, das nicht am Treffen teilnimmt, kann vorab die Artikel, die es begutachtet hat mit den entsprechenden Gutachtern diskutieren. Die im OCS konfigurierbare Anonymität der Gutachter während des Begutachtungsprozesses, spiegelt eine Lösung des Pattern *Consensus on PC Papers* wider. Kombiniert mit dem *double-blind review* können Artikel von PC-Mitgliedern begutachtet werden. Die PC-Mitglieder erfahren nicht, wer der Autor des Artikels ist, noch wer die Gutachten geschrieben hat. Der gesamte Begutachtungsprozess, inbegriffen die Diskussionen, erfolgt anonym. Dies hat zur Folge, dass keine Ausnahmeregelungen für die PC Einreichungen notwendig sind und alle Artikel eine Gleichbehandlung erfahren.

Bedingt durch die flexiblen Features des OCS ist der Ansatz aus dem Artikel *Identify the Champion* [Nie00] mit dem OCS ebenso umsetzbar, wie die Anforderungen verschiedener Kommunities. Sind Anforderungen von Kommunities aus mehreren Fachbereichen, wie zum Beispiel der Informatik, Medizin oder Erziehungswissenschaften zu erfüllen, ist die Agilität des System von sehr großer Bedeutung, da die Ansprüche an die Systemfunktionalitäten differieren. Diese Art von (frühzeitiger) Bestimmung von Befürwortern findet nicht in allen Kommunities ihre Anwendung. Fakt ist, das Kommunities unterschiedlichste Auffassungen darüber haben, wie für ihre Konferenz der Begutachtungsprozess zu erfolgen hat. Es haben sich Vorgehensweisen in den Kommunities gefestigt, die aus der Sicht der jeweiligen Konferenz den optimalen Begutachtungsprozess darstellt. Eine Flexibilität des Begutachtungssystems in Bezug auf die Anpassbarkeit des Begutachtungsprozesses, ist daher aus mehrjähriger Erfahrung essentiell wichtig.

Teil II

Verwendete Technologien

Kapitel 3

Frameworks

Dieses Kapitel stellt das Entwicklungswerkzeug ABC und das für die Entwicklung von web-basierten Applikationen implementierte ABC-Module *Extended Web Info Service* (EWIS) vor.

3.1 ABC/jABC

Das 1995 von der Firma METAFrame entwickelte *Agent Building Center* (ABC) [SM99] ist eine graphbasierte Entwicklungsumgebung, die den Prozess der Umsetzung von Softwareprojekten effektiv unterstützt und das Arbeiten einer Gruppe von Entwicklern vereinfacht. Mittlerweile wurde das ABC von dem in Java programmierten jABC abgelöst, das gänzlich losgelöst von der Programiersprache C++ ist und Funktionalitäten als Plugins zur Verfügung stellt. Neben dem *Springer-OCS* Projekt wurden die Frameworks unter anderem erfolgreich für die Projekte *Electronic Tool Integration* (ETI) [MBK97, Bra01, Mar05, MRSL07], CTI [1] [Nie03, MSR05], *Teaching Management Platform for Universities* (TEMPLUS) [Gso04] und IKEA [HMM+06] eingesetzt.

Das ABC basiert auf dem *Lightweight Process Coordination* Ansatz [MS04b], der das bekannte *Model View Controller* (MVC) Konzept um eine Schicht erweitert. Diese Koordinationsschicht spiegelt die graphische Modellierungsebene des ABC wider und ermöglicht eine abstrahierende und flusszentrierte Sichtweise auf den Geschäftsprozess. Auf diesem erweiterten Konzept baut das *Aggressive model-driven development* (AMDD) [MS04a] auf. Das AMDD versetzt Personen ohne tiefere Kenntnisse über die Diensttechnik oder Program-

[1] Computer Telephony Integration

miersprache in die Lage, Geschäftsprozesse zu modellieren oder bestehende zu modifizieren. Hierbei handelt es sich um die Applikationsexperten, welche nachstehend genauer spezifiziert werden.

Bei dem Entwicklungsprozess von Web-basierten Applikationen, wie es beim OCS der Fall ist, sind verschiedene Aufgaben zu erfüllen, die unterschiedlichste Qualifikationen voraussetzen. Die an einer Projektdurchführung beteiligten Personen lassen sich in die folgenden Rollen klassifizieren:

- **System-Ingenieur** - Analysiert die Aufgabe und erstellt Lösungsansätze, die mit dem Kunden diskutiert und zur Umsetzung freigegeben werden. Er ist für die Erfüllung des spezifizierten Lösungsansatzes und dessen Integration verantwortlich.
- **OO-Programmierspezialist** - Design und Implementierung der OO-Business-Objekte. Er ist verantwortlich für die Business-Objekte, die als Basis für die SIBs der Anwendung dienen.
- **SIB-Entwickler** - Sein Aufgabenbereich besteht aus der Implementierung der für die Anwendung benötigten SIBs unter Verwendung der Software-Komponenten des OO-Spezialisten.
- **Applikationsexperte** - Spezifiziert, modelliert mit Hilfe der vom SIB-Entwickler implementierten SIBs die Geschäftsprozesse als SLGs.
- **GUI-Designer** - Erstellt die Bedienungsoberfläche der Applikation, die z.B. in der Sprache HTML [2], JSP [3] oder JSF [4] realisiert wird.

Mit diesem Konzept kann die Entwicklung einer Applikation strukturiert auf die vorgestellten Rollen aufgeteilt und dadurch eine Trennung der Verantwortlichkeiten erzielt werden (vgl. [MS06, Mar07]). Der graphbasierte Ansatz (siehe Kapitel 3.1.1) vereinfacht das Design und die Modellierung des Kontrollflusses. Der Applikationsexperte benötigt daher keine IT-Kenntnisse, um einen Dienst zu modellieren bzw. in Bezug auf sich ändernde Geschäftsprozesse anzupassen. Der modellierte Kontrollflussgraph visualisiert den Workflow der Applikation und dient somit bereits zur Designphase als Grundlage zur Diskussion des Projektes mit dem Geschäftspartner (Kunden). Der Kontrollflussgraph stellt im Prinzip ein Aktivitätsdiagramm dar, wie es in der Modellierungssprache UML [5] [Fow03] bekannt ist.

[2] Hypertext Markup Language
[3] JavaServer Pages
[4] Java Server Faces
[5] UML steht für Unified Modelling Language und dient für die Anforderungsanalyse von

3.1.1 Modellierung und Validierung

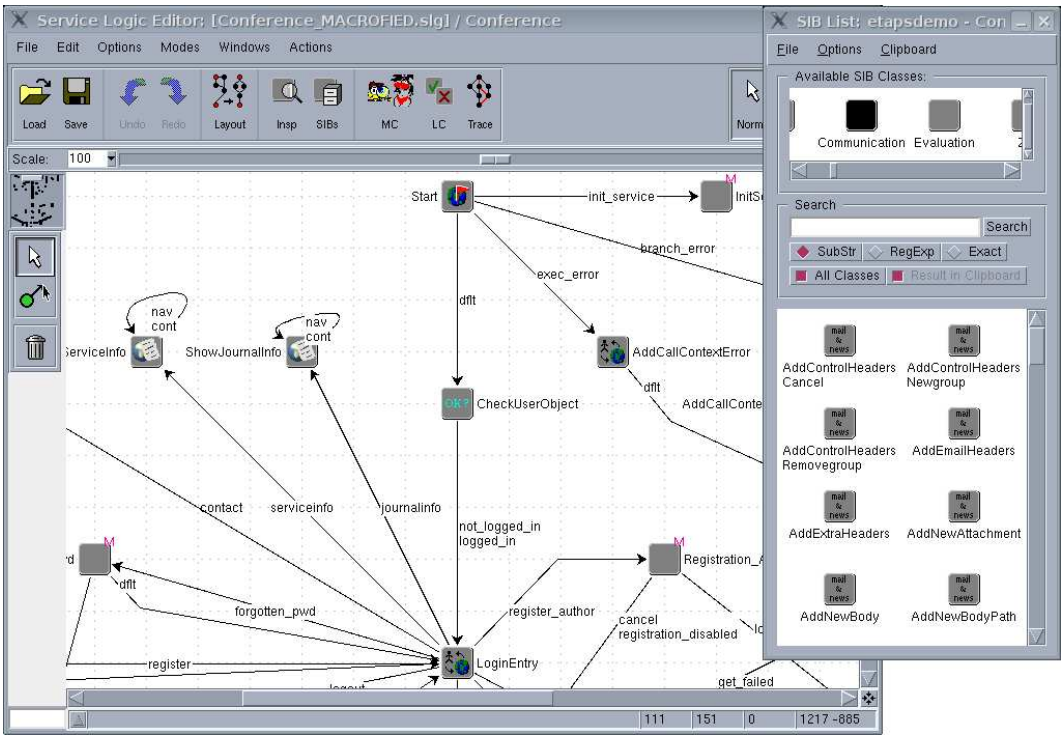


Abbildung 3.1: Das ABC mit geladenem OCS-SLG und SIB-Palette

Die Geschäftslogik wird mittels des ABC's durch gerichtete Kontrollflussgraphen modelliert. Abbildung 3.1 zeigt das ABC mit geöffnetem Service Logic Graph des OCS und die dem Projekt zugehörige SIB-Palette. Die *Service Independent Building Blocks* (SIBs) sind Softwarebausteine mit einem einfachen Aufbau. Ein SIB ist durch zwei wesentliche Merkmale gekennzeichnet. Da ist zum einen die Spezifikation des SIBs, welche die notwendigen Informationen für die Modellierungsebene beinhaltet und zum anderen die Implementierung des SIBs. Die Implementierung der SIBs ist für den OCS in Java geschrieben und definiert die Aktionen, die ein aufgerufener SIB durchführt, wie z.B. das Lesen von Benutzerdaten aus einer Datenbank.

Ein gerichteter Kontrollflussgraph besteht aus einer beliebigen Anzahl von Knoten, die mittels Kanten verbunden werden. Die Knoten entsprechen den

Anwendungen mittels verschiedener Diagrammtypen, wie Aktivitätsdiagramme, Zustandsdiagramme, Klassendiagramme, etc.

in Abbildung 3.1 verwendeten SIBs und die Kanten den verbindenden Pfeilen, die jeweils von einem zum anderen SIB zeigen und so den Kontrollfluss bestimmen. Ein weiteres Modellierungselement ist das Makro, welches Kontrollflussgraphen oder andere Makros enthält. Es dient zur Kapselung von Teilgraphen, womit der Umgang mit oft verwendeten Graphen vereinfacht und zentralisiert wird. Die in Abbildung 3.1 gezeigte SIB-Palette stellt eine SIB-Bibliothek dar, die neben den spezifischen SIBs der Applikation eine große Vielfalt von bereits implementierten und durch den Einsatz in verschiedenen Projekten erprobten und somit robusten Bausteinen zur Verfügung stellt.

SIBs sind wiederverwendbare Bausteine, die nicht nur in der eigenen Domäne, für die sie ursprünglich entwickelt wurden, sondern auch Domänen-übergreifend zum Einsatz kommen. Viele SIBs sind parametrisierbar und somit flexibel einsetzbar.

Das im ABC erstellte Modell der Geschäftslogik ist entkoppelt von der darunter liegenden Implementierung der eigentlichen Funktionalitäten der Applikation. Insbesondere ist dieses gerichtete Graphmodell bereits zur Designzeit validierbar mittels formaler Methoden, z.B. mittels Model Checking, wodurch festgelegte Policies, Regeln und Bedingungen frühzeitig in der Designphase des Kontrollflusses überprüft werden können, ohne das es einer Implementierung der SIBs bedarf.

Als temporale Logik kommt CTL [6] [CGP01] zum Einsatz. Spezifikationen in Form von CTL-Formeln können mit dem FormulaBuilder [JMS06] graphisch im ABC modelliert werden. Die graphbasierten Formeln werden automatisch nach CTL überführt und dem Model Checker GEAR [BMRS07a, BMRS07b] übergeben. GEAR übersetzt die CTL-Formel ins μ-Kalkühl [EJS93] und führt die Validierung gegen das Graphmodell durch.

Die Computation Tree Logic, welche für die Beschreibung der gewünschten Systemeigenschaften verwendet wird, wird nachstehend genauer vorgestellt.

Sei AP eine Menge von atomaren Propositionen, wie z.B. die Annotationen an den Knoten des Graphmodells, die die Eigenschaften der Komponenten (SIBs) des SLGs beschreiben.

Definition 3.1.1 ***CTL Syntax***
Für $p \in \mathsf{AP}$ *sind die CTL Formeln definiert als:*

$$\phi ::= p \mid \neg\phi \mid \phi \vee \phi \mid \mathsf{EX}[\phi] \mid \mathsf{E}[\phi\ \mathsf{U}\ \phi] \mid \mathsf{A}[\phi\ \mathsf{U}\ \phi]$$

[6] Computation Tree Logic

Die Semantik einer CTL-Formel wird definiert relativ zu einem KTS [7], ein Transitionssystem mit Aktionsbezeichnungen an den Kanten und atomare Propositionen an den Knoten. Pfade in einem KTS repräsentieren die möglichen Ausführungssequenzen eines Systems.

Definition 3.1.2 ***Pfad***
Ein unendlicher Pfad ist eine Sequenz von Zuständen $s_0, s_1, s_2, \cdots$, so dass gilt $(s_i, a, s_{i+1}) \in \rightarrow$ für $a \in \mathsf{Act}$. S^ω kennzeichnet die Menge aller unendlichen Pfade. Der $(i+1)$-te Zustand auf einem Pfad $\pi \in S^\omega$ wird mit $\pi[i]$ bezeichnet.

$P_K(s) = \{\pi \in S^\omega | \pi[0] = s\}$ bezeichnet die Pfade die vom Zustand s im Kripke Transitionssystem K starten.

Die Semantik von CTL-Formeln wird induktiv über ihren Aufbau definiert:

Definition 3.1.3 ***CTL Semantik***
Sei $K = (S, \mathsf{Act}, \rightarrow, I)$ ein Kripke Transitionssystem mit atomaren Propositionen AP, $p \in \mathsf{AP}$ ist eine atomare Proposotion, $s \in S$ ist ein Zustand und ϕ, ψ sind CTL Formel. Die Erfülltheitsrelation $\models$ wird definiert durch:

$s \models p$	$gdw\ p \in I(s)$
$s \models \neg\phi$	$gdw\ s \models \phi$ *nicht erfüllt ist*
$s \models \phi \vee \psi$	$gdw\ s \models \phi$ *oder* $s \models \psi$
$s \models \mathsf{EX}[\phi]$	$gdw\ \exists\ \pi \in P_K(s).\pi[1] \models \phi$
$s \models \mathsf{E}[\phi\ \mathsf{U}\ \psi]$	$gdw\ \exists\ \pi \in P_K(s).\exists\ j \geq 0.\pi[j] \models \psi \wedge$ $\forall\ 0 \leq k < j.\pi[k] \models \phi$
$s \models \mathsf{A}[\phi\ \mathsf{U}\ \psi]$	$gdw\ \forall\ \pi \in P_K(s).\exists\ j \geq 0.\pi[j] \models \psi \wedge$ $\forall\ 0 \leq k < j.\pi[k] \models \phi$

Weitere Operatoren können abgeleitet werden:

- $\phi \Rightarrow \psi \equiv \neg\phi \vee \psi$ und $\phi \wedge \psi \equiv \neg(\neg\phi \vee \neg\psi)$
- $\mathsf{AX}[\phi] \equiv \neg\mathsf{EX}[\neg\phi]$
- $\mathsf{EF}[\phi] \equiv \mathsf{E}[\text{true}\ \mathsf{U}\ \phi]$ und $\mathsf{AF}[\phi] \equiv \mathsf{A}[\text{true}\ \mathsf{U}\ \phi]$
- $\mathsf{EG}[\phi] \equiv \neg\mathsf{AF}[\neg\phi]$ und $\mathsf{AG}[\phi] \equiv \neg\mathsf{EF}[\neg\phi]$

[7] Kripke Transition System

Neben den üblichen logischen Konektoren $\neg$ und $\vee$ enthält CTL die temporalen Operatoren X und U. Formeln mit diesen Operatoren werden durch einen Pfadquantor E oder A eingeleitet, der anzeigt ob die Formel auf einem oder allen ausgehenden Pfaden gelten soll.

- X (next-time) bezieht sich auf den nächsten Zustand
- F (finally) bezieht sich auf den Zustand der eventuell erreicht wird
- G (generally) bezieht sich auf den gesamten Pfad
- ϕ U ψ (until) ϕ muss solange gelten bis ψ gilt, oder ψ gilt direkt ohne vorheriges Vorkommnis von ϕ

Wir interpretieren SLGs als Kripke Transitionssysteme, wobei die SIBs eines SLGs die Knoten und deren Verzweigungen die Kanten des KTS sind. Die atomaren Propositionen in den Formeln können eine Existenz von einem SIB selbst, die Werte von SIB-Parametern und die Erreichbarkeit von SIBs/Diensten über den Verzweigungen sein.

Der temporale Operator DIAB (*diamond backward*) verhält sich wie ein EX Operator mit umgekehrtem Transitionspfeil. Wir nennen dieses Verhalten EXB, das besagt, dass mindestens ein Vorgänger SIB (Knoten) existiert mit dem Namen, der rückwärts über einen branch (Kante) mit der Benennung erreichbar ist.

3.1.2 Ausführbare Geschäftsprozesse

Mittels des ABCs und eines integrierten Code-Generators lassen sich Graphmodelle, wie in Abbildung 3.1 dargestellt, automatisiert in eine Programmierungssprache überführen. Im Falle des OCS, der in Java implementiert ist, wird für das Modell automatisch eine Java-Implementierung erstellt. Die Vorgehensweise entspricht der einer Adjazenzliste, die von einer Laufzeitumgebung interpretiert und ausgeführt wird (siehe Kapitel 3.2). Abbildung 3.2 zeigt einen Ausschnitt einer Java-Klasse, die einen Kontrollflussgraphen beschreibt. Das Modell wurde auf die durch einen Pfeil in Beziehung stehenden SIBs heruntergebrochen. So wird z.B. der graphische Zusammenhang zwischen den SIBs *CreateBody* und *SendAndStoreMimeMessage* als `addBranch (‘‘CreateBody’’, ‘‘created’’, ‘‘SendAndStoreMimeMessage’’);` definiert und der leichtgewichtigen Prozess-Engine EWIS übergeben. Der Kontrollfluss bestimmt sich aus der Position der Argumente der Methode. Somit wird von *CreateBody* über die Kante *created* zu *SendAndStoreMimeMessage* gegangen.

```
package de.metaframe.ewis.application.editorial.services;

/*
 * Generated automatically by Web Info Service Compilation utility
 *  of the Agent Building Center.
 */
public class ConferenceLogic extends de.metaframe.ewis.Logic
{
   public ConferenceLogic ()
       throws javax.servlet.ServletException
   {
      super ();
      addBranch ("AccessDBEntryInfo", "dflt", "VectorContains_lockedUsers");
      addBranch ("CreateBody", "created", "SendAndStoreMimeMessage");
      addBranch ("OpenConnection", "connected", "GetAllInvolvedForumUsers");
      addBranch ("PreparePostingNotification", "not_send", "LoopOverUserlist");
      addBranch ("PreparePostingNotification", "send", "FillTemplate");
      addBranch ("FillTemplate", "filled", "CreateNewMimeMessage");
      addBranch ("VectorContains_lockedUsers", "false", "CheckPostingNotification");
      addBranch ("VectorContains_lockedUsers", "true", "LoopOverUserlist");
      addBranch ("CloseConnection", "closed", "NOOP2");
      addBranch ("AddEmailHeaders", "added", "CreateBody");
      addBranch ("SendAndStoreMimeMessage", "not_delivered", "LoopOverUserlist");
      addBranch ("SendAndStoreMimeMessage", "partially_delivered", "LoopOverUserlist");
      addBranch ("SendAndStoreMimeMessage", "delivered", "LoopOverUserlist");
      addBranch ("CreateNewMimeMessage", "created", "AddEmailHeaders");
      addBranch ("GetAllInvolvedForumUsers", "ok", "LoopOverUserlist");
      addBranch ("SetUserPostedFlag", "dflt", "OpenConnection");
      addBranch ("LoopOverUserlist", "next", "AccessDBEntryInfo");
      ...
```

Abbildung 3.2: Generierter Java Quellcode der Geschäftslogik.

Alternativ könnte die Adjazenzliste bei kurzlebigen und langlebigen Prozessen auch in der Datenbank verwaltet werden. Um Leistungseinbußen bei kurzlebigen Prozessen, wie es bei der Navigation von Web-Applikationen der Fall ist, zu umgehen, müsste *Prefetching* betrieben werden, d.h. die Adjazenzliste muss im Speicher vorgehalten werden.

Zusätzlich zu der Java-Abbildung des Workflows wird eine weitere Java-Klasse generiert, die die Instanziierungen der in dem Graphmodell eingesetzten SIBs definiert. Die für die Modellierung geschriebenen SIB Spezifikationen sind parametrisierbar, d.h. SIBs sind wiederverwendbar und Domänen übergreifend einsetzbar. Die Werte der definierten Parameter werden an die SIB-Implementierung als Argumente des Konstruktors übergeben. In Abbildung 3.3 ist ein Ausschnitt des Inhaltes einer solchen generierten Klasse aufgezeigt. Wird der SIB *CreateBody* aus dem Beispiel zuvor betrachtet, so ist dieser in Abbildung 3.3 in der Zeile `addSIB ‘‘CreateBody’’, new de.metaframe.ewis.application.mailmgmt.sib.CreateBody ( ‘‘filledbody’’, ‘‘message’’));` wiederzufinden. Spezifiziert wird einerseits der Klassenname des SIBs und der Aufruf der Klasse mit den im Modell

gesetzten Parametern, was zur Instanziierung zur Laufzeit benötigt wird.

```
package de.metaframe.ewis.application.editorial.services;

/*
 * Generated automatically by Web Info Service Compilation utility
 *  of the Agent Building Center.
 */
public class ConferenceSIBContainer extends de.metaframe.ewis.sib.SIBContainer
{
   public ConferenceSIBContainer ()
       throws javax.servlet.ServletException
   {
      super ();
      addSIB ("OpenConnection",
              new de.metaframe.ewis.application.mailmgmt.sib.OpenConnection
              ("mailcommconnectionuser", "mailtransportsession","mailconnection1"));
      addSIB ("CreateBody",
              new de.metaframe.ewis.application.mailmgmt.sib.CreateBody
              ("filledbody", "message1"));
      addSIB ("CheckFeaturePermitted", new CheckFeaturePermitted ("F-FORUM"));
      addSIB ("CalculateForbiddenNewsgroups", new CalculateForbiddenNewsgroups ());
      addSIB ("CheckUserProfile",
              new CheckOrSetUserState ("check", "session", "editorialuser",
                                       "incompleteProfile", "yes"));
      addSIB ("IncompleteProfileBranch", new IncompleteProfileBranch ());
      addSIB ("GetNavbarFeatures", new GetNavbarFeatures ());
      addSIB ("SwitchActiveNavbarButton", new SwitchActiveNavbarButton ("forums"));
      addSIB ("AddCallContextError",
              new de.metaframe.ewis.application.sib.AddCallContextError ("invalid_ticket"));
      addSIB ("UserLockedHandling", new UserLockedHandling ());
      ...
```

Abbildung 3.3: Generierter Java Quellcode zur Instanziierung der SIBs.

3.2 EWIS

Die *Extended Web Info Service* (EWIS) Plattform erweitert das ABC um Funktionalitäten, die für die Entwicklung von Web-basierten Applikationen benötigt werden. Dazu gehören unter anderem die Projekte *ewis_base* und *ewis_comm*. Ewis_comm gibt den Applikationsexperten eine vollständige SIB-Bibliothek für die Modellierung von Email- oder News-Kommunikation an die Hand, die von mir konzipiert und entwickelt wurde. Ewis_base hält eine Vielzahl von implementierten und validierten Basis-SIBs bereit, die in drei SIB-Gruppen eingeteilt werden.

- **Interaktion-SIBs:** Zu dieser Gruppe gehören die SIBs, die z.B. eine HTML-Seite anzeigen, die sogenannten ShowSIBs. Sobald im Kontrollfluss ein ShowSIB aufgerufen wird, hält der Kontrollfuss an und wartet

auf eine Interaktion des Dienstbenutzers. Sobald eine Aktion ausgeführt wurde, wird der Kontrollfluss entsprechend weiter abgearbeitet bis der nächste ShowSIB erreicht wird.

- **Aktion-SIBs:** Diese Art von SIBs werden nacheinander abgearbeitet, sie befinden sich i.d.R. im Graphen zwischen zwei ShowSIBs. Die SIBs lesen Daten aus der Datenbank, bereiten diese auf oder verarbeiten eingehende Daten.
- **Transfer-SIBs:** EWIS stellt mehrere Datenkontexte zur Verfügung zwischen denen unter Verwendung der Transfer-SIBs Objekte kopiert werden können.

EWIS verfügt über eine eigene Laufzeitumgebung mit einer leichtgewichtigen Prozess-Engine, die auf Basis der generierten Java-Klasse des Graphmodells den Workflow der Applikation verwaltet und steuert. EWIS baut auf der Servlet-Technologie [mic09f] auf, d.h. dass die Instanzen der Java-Klassen in einem Web-Container ausgeführt werden, wie es bei dem Web-Server Tomcat [Fou09c] bzw. JBoss [Hat09] der Fall ist. Des Weiteren verwaltet EWIS mehrere Datenkontexte, die es ermöglichen, die Daten zwischen den einzelnen SIBs zu transferieren. Die Hauptkontexte werden nachstehend aufgezeigt.

- **Container Context:** Gespeicherte Objekte sind aus allen Servlets einer Applikation zugänglich.
- **Service Context:** Auf den Service Context können alle Sessions des Services zugreifen.
- **Session Context:** Eine Session ist nur einem Dienstbenutzer zugeordnet. Sich in dem Session Context befindliche Objekte sind nur von der selben Session aus zugänglich.
- **Call Context:** Der Call Context existiert immer nur zwischen zwei Interaktion SIBs und dient zur Thread-Sicherheit.

EWIS wurde in Bezug auf den OCS und seiner Dienstfamilien unter anderem durch eine neue Softwarekomponente für das Verwalten von Multipart-Uploads [8] und einer Encoding-Komponente [9] erweitert.

[8] Das Hochladen von Texten und Dateien in einer Anfrage (Request).

[9] Es werden Sonderzeichen für die richtige Darstellung in spezifische HTML-Zeichen kodiert.

Kapitel 4

Realisierte Konzepte

Dieses Kapitel stellt die entwickelten Konzepte, die bereits erfolgreich in Projekten des OCS eingesetzt wurden, vor. Zu den primären Ansätzen gehören der Feature-orientierte Entwicklungsansatz und das zur Laufzeit rekonfigurierbare Rollen/Rechte Management, die zum Erfolg des OCS maßgeblich beigetragen haben. Die Features *Bidding- und Verteilungsmechanismus* und *Konfigurierbarer Bewertungsworkflow* werden detaillierter betrachtet, da sie repräsentativ sowohl zur Vereinfachung der Arbeitsabläufe für den Konferenzleiter beigetragen haben, als auch die Flexibilität des Systems und die Wiederverwendbarkeit darstellen.

4.1 Feature-orientierter Ansatz

Es gibt unterschiedliche Herangehensweisen für die Entwicklung einer Webbasierten Applikation. In diesem Kapitel wird der Feature-orientierte Ansatz beschrieben, der für die Umsetzung des OCS verwendet wurde. Der OCS ist ein Beispiel für die Entwicklung einer komplexen und rekonfigurierbaren Webbasierten Anwendung. Damit die Komplexität eines Softwaresystems während der Entwicklungsphase und späteren Pflege beherrschbar bleibt, werden Dienstfunktionalitäten als Kontrollflussgraphen modelliert und zu Features zusammengefasst. Dieses Vorgehen bestimmt einen strukturierten und validierbaren Entwicklungsansatz für die Umsetzung von komplexen Software-Projekten. Dienstfunktionalitäten werden gekapselt und sind im selben Dienst sowie in der Produktlinie wiederverwendbar [Cza04]. Kombiniert mit dem MDD [1] auf Graphebene des ABC können auf einfache Weise Services, die die Dienstfunktiona-

[1] Model-Driven Development

litäten (Features) der Produktlinie repräsentieren, zu einem Dienst orchestriert werden, was wiederum dem SOA Gedanken entspricht. Hierarchien nehmen hier eine zentrale Rolle ein, die es ermöglichen, Applikationen auf verschiedenen Abstraktionsebenen zu modellieren ohne dabei die Ausführbarkeit zu verlieren. Bei BPEL wurde erst spät erkannt [KKL+05], dass eine Hierarchie für komplexe Prozessketten bzw. Geschäftsabläufe für die Übersichtlichkeit, Verständlichkeit und vor allem für die Wiederverwendbarkeit von entscheidender Bedeutung sind. MDA [2] ist ein Framework für MDD [SVC06], was dem Entwickler ermöglicht Applikationen zu spezifizieren ohne sich auf Technologien festzulegen. Die Herausforderung ist es, solche Techniken in einfacher Form dem Anwendungsexperten zugänglich zu machen, um Geschäftsprozesse auch in Hinblick auf Produklinien zu modellieren. Der in [BM09] vorgestellte Ansatz auf Basis von UML und MDA verwendet nicht die Abstraktionsebene für das Verständnis von Anwendungsexperten.

4.1.1 Feature Beschreibung

Im Folgenden wird der Begriff `Feature` eingeführt, der eine zentrale Rolle für die Beschreibung des Entwicklungskonzeptes und das später zu erläuternde Rollen/Rechte Management einnimmt. Der Begriff `Feature` wurde erstmalig durch den technischen Report [KCH+90] definiert und ist heutzutage sehr gebräuchlich, wenn es sich um die Spezifikationen von Leistungsmerkmalen einer Software oder Hardware handelt. Gerade im Bereich des Marketing und der Werbung werden potentielle Kunden von Produkten mit dem Begriff Feature konfrontiert, an denen Produktlinien definiert werden können.

In dieser Arbeit wird ein Feature wie folgt definiert:

- Ein Feature ist eine Teilfunktionalität eines Gesamtsystems.
- Jedes Feature erweitert den Funktionsumfang des Gesamtsystems.
- Ein Feature kann andere Features inkludieren bzw. von anderen abhängen.
- Features werden bestimmt durch externe Sichten auf das System, wie die eines Benutzers/Kunden oder Dienstanbieters.
- Die Granularität wird bestimmt durch Marketing- oder Provisioning-Zwecke.

[2] Model-Driven Architecture

Im OCS werden Dienstfunktionalitäten in Features klassifiziert. Solche Dienstfunktionalitäten können einerseits dienst-spezifische Geschäftsprozesse, wie das Einreichen von Papieren, andererseits die Zugriffe auf Sourcen oder verwendete Dienste sein, wie CVS, Mail oder Datenbanken.

Features werden Rollen zugeordnet, die wiederum einem Dienstbenutzer zugewiesen sind und diesem Zugriff auf die entsprechenden Dienstfunktionalitäten geben. Die Rollen können fein-granular konfiguriert und dadurch konferenzspezifische Anforderungen realisiert werden. Eine derartige spezifische Anforderung könnte der zweistufige Begutachtungsprozess sein, der es den Gutachtern ermöglicht, die Begutachtungsaufgabe an Reviewer weiterzuleiten, um deren Gutachten in das eigene einfließen zu lassen.

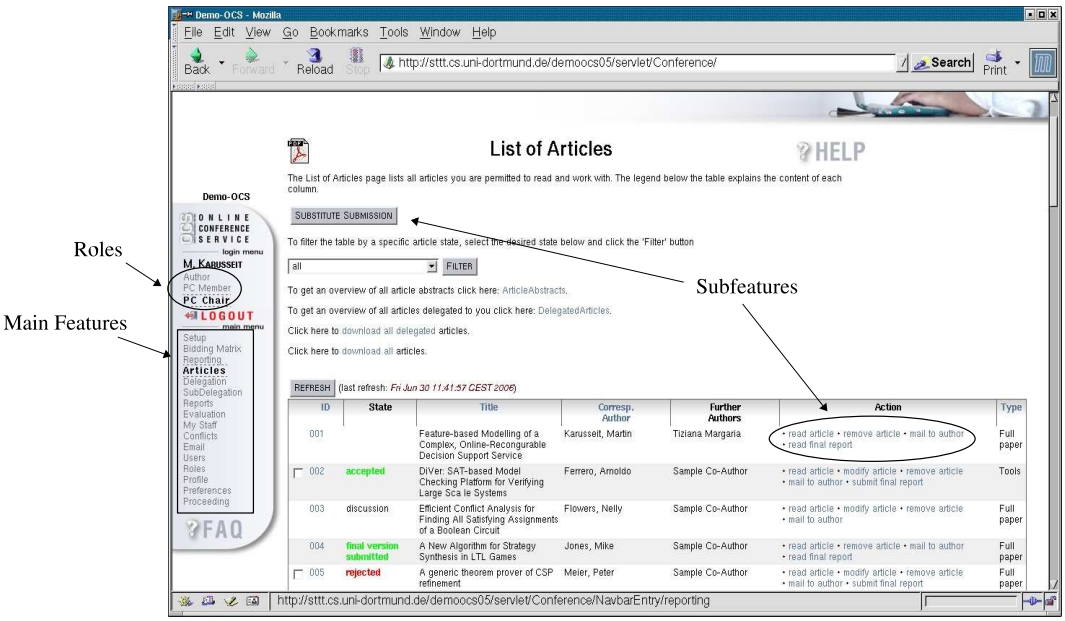


Abbildung 4.1: Features im OCS: Benutzersicht

Abblidung 4.1 vermittelt einen Überblick über die Verwendung der Features und deren Hierarchie im OCS aus der GUI-Sicht eines Dienstbenutzers. Ist ein Benutzer im Dienst angemeldet, so sieht dieser die Navigationsleiste auf der linken Seite und die Informationsseite der Applikation, die die eigentlichen Informationen enthält, rechts daneben. Mittels der Navigationsleiste ist es einerseits möglich, zwischen den Rollen des Benutzers oder zwischen den Haupt-Features zu wechseln. Hierzu zählen unter anderem die Features Articles, Setup, Users, Roles, etc. Der momentane Benutzer hat mit seiner selektierten Rolle `PC Chair` das Hauptfeature `Articles` gewählt, das graphisch auf der

Seite hervorgehoben ist. Die zugeordnete Informationseite zeigt die Hauptseite des Article Features an, die `List of Articles` Seite. Diese Seite beinhaltet Links zu einigen Sub-Features von Articles, wie *Substitute Submission*, *read article* und *remove article*. Sub-Features sind kleine Funktionalitäten die im Zusammenspiel mit anderen Sub-Features ein Haupt-Feature definieren.

4.1.2 Feature-basiertes Design

Im Feature-basierten Entwicklungsansatz wird das Design einer Applikation in einzelne Schichten unterteilt. Die Features werden hierarchisch in Beziehung gesetzt, wobei die Umsetzung in Komponenten und in Bezug auf den Kontrollflussgraphen des ABC in Makros realisiert ist. Eine Verschachtelung von Makros, und somit der Feature des Dienstes, zeigt die Wiederverwendbarkeit und die Beziehungen zwischen den Features auf. Jedes Makro ist für sich genommen ein SLG, der die Leistungsmerkmale eines Features modelliert. Die Realisierung eines Features wird verwirklicht durch die Kombination von SIBs und Makros in SLGs und in Makros selbst.

Das mit dieser Vorgehensweise konzipierte Entscheidungssystem OCS umfasst mittlerweile eine SLG-Struktur mit über 2500 Knoten und 3500 Kanten. Das Entscheidungssystem ist von seinem Implementierungsumfang, beruhend auf der großen Anzahl der Features und deren Flexibilität, ein umfangreiches Softwaresystem. Derartig große Softwareprojekte bringen von *Haus aus* eine große Komplexität mit sich, wodurch die Überschaubarkeit des Projektes abnimmt. Mittels des hier vorgestellten Feature-basierten Ansatzes werden Funktionalitäten gekapselt und graphbasiert in Beziehung gestellt, was zu einer Strukturierung und Überschaubarkeit der Dienstkomponenten führt. In Abbildung 4.2 ist der Hauptgraph des OCS dargestellt, der durch seine abstrakte Sicht auf die Applikation recht einfach und übersichtlich ist.

Dieser SLG visualisiert die oberste Kontrollflussschicht der Anwendung und setzt deren Hauptfunktionalitäten (Haupt-Features) mittels eines gerichteten Graphen in Beziehung. Grundlegend lässt sich dieser SLG in die für eine personalisierte Web-Anwendung klassischen drei Bereiche unterteilen:

1. **Initialisierung:** Während der Initialisierung des Dienstes wird das Makro *InitService* durchlaufen. Dieses Makro beinhaltet den Initialisierungskontrollfluss z.B. für die Instanziierung von Geschäftsobjekten und Datenbankadministratoren sowie für die Registrierung an Diensten wie Datenbank, News, Email oder CVS. Des Weiteren werden Dienst-spezifische Konfigurationsdaten, wie z.B. der Name der Anwendung, Kontaktadres-

sen oder das Aussehen der GUI, in Dienst-Kontexte hinterlegt. Ob die Initialisierung direkt beim Starten des WebServers oder erst beim ersten Aufruf des Dienstes geschehen soll, wird im Deployment-Descriptor [3] der Anwendung spezifiziert.

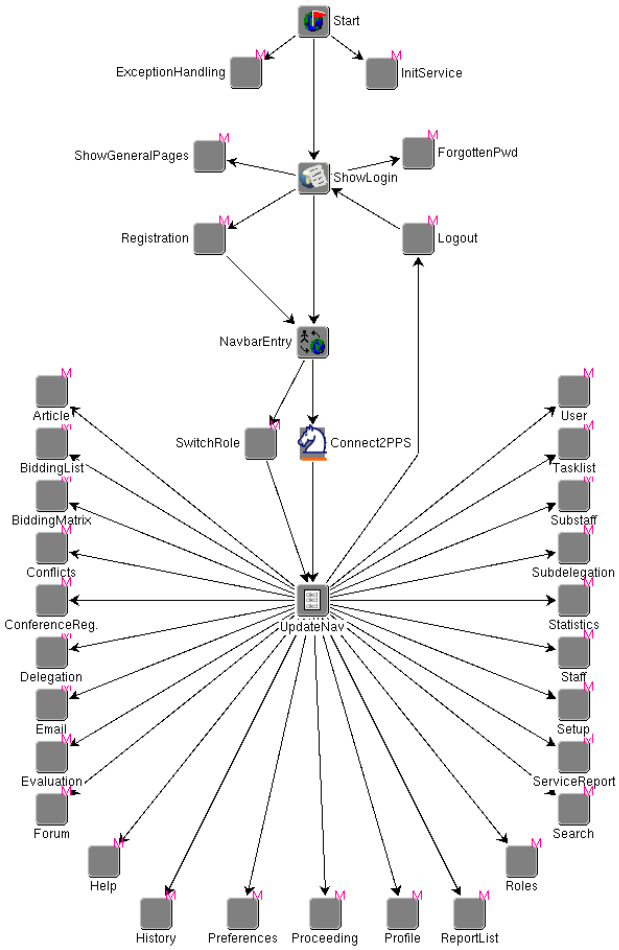


Abbildung 4.2: Struktur eines Hauptgraphen: Gezeigt am Beispiel des OCS

2. **Öffentlicher Bereich:** Im Öffentlichen Bereich befinden sich die Features, auf die ohne eine vorherige Authentifizierung zugegriffen werden darf. Hierzu gehören die Features *Registration*, *ForgottenPwd* und *ShowGeneralPages* die als Makros im SLG wieder zu finden sind. Im Makro *ShowGeneralPages* sind allgemeine Dienstseiten gekapselt, wie z.B. die Informationen zu dem Dienst oder Kontaktinformationen.

[3] Eine auf Servlet-basierende Web-Anwendung benötigt einen Deployment-Deskriptor, eine XML-basierte Datei, unter anderem für die Konfiguration eines Servlets.

3. **Interner Bereich:** Nur registrierte Benutzer können sich in den internen Bereich einloggen. Nach erfolgreicher Authentifizierung stehen dem Benutzer in Abhängigkeit seiner Zugriffsrechte die Features (Makros) im Graphen zur Verfügung, auf die eine Kante zeigt, die vom SIB *UpdateNav* ausgeht, wie z.B. das *Article*, *User* oder *Logout* Feature.

Wie in Abbildung 4.2 gezeigt, ist jedes Feature auf Graph-Ebene durch ein Makro realisiert, das alle Funktionalitäten und Arbeitsabläufe des jeweiligen Haupt-Features repräsentiert. Die Abbildung 4.3 stellt das geöffnete Makro bzw. das Graphmodell von dem *Article* Feature vor.

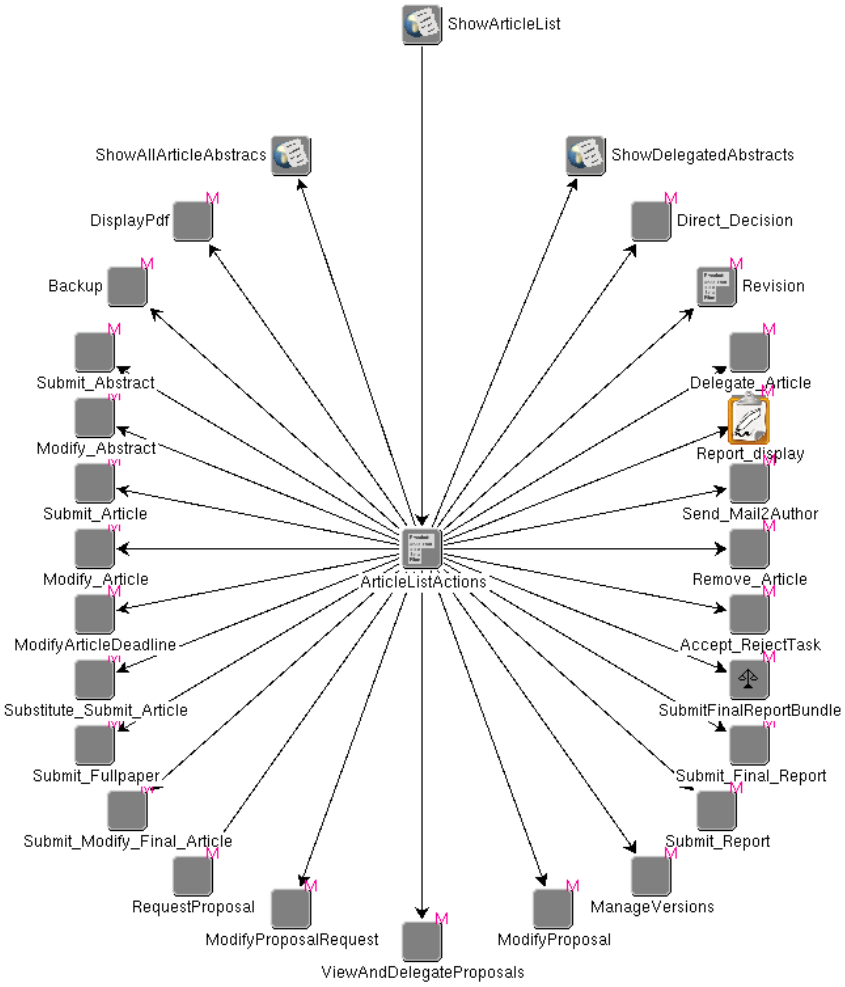


Abbildung 4.3: SLG des Article Feature: Hierarchische Struktur des Feature

Zu sehen ist ein SLG der aus den Elementen SIBs, den Sub-Makros und den Koordinationselementen (hier Kanten) modelliert wurde. Aus der Dienstbenut-

zersicht finden sich die Haupt-Features in der Navigationleiste und die Sub-Features in der Kontextseite wieder, siehe Abbildung 4.1.

In Abhängigkeit der zu konzipierenden Web-Applikation können Features in feingranulare Sub-Features unterteilt werden, die selbst als SLGs modelliert sind, wie in Abbildung 4.3 für das Haupt-Feature *Article* zu sehen. Das Sub-Feature besteht aus einem einfachen Kontrollfluss und koordiniert Aufrufe in Bezug auf die Sub-Features. Die Verfeinerungstiefe der Stufen ist nicht vorgegeben, wodurch jedes Sub-Feature ebenfalls durch Makros strukturierbar ist.

Die Feature-übergreifende Verwendung ist erlaubt und erwünscht, um einerseits ein hohes Maß an Wiederverwendbarkeit zu erzielen, andererseits Redundanzen von Funktionalitäten, sprich Implementierungen, zu vermeiden. An dem *Article* Feature ist die Benutzung von Sub-Features aus anderen Haupt-Features zu erkennen. Gehören z.B. die Sub-Features *SubmitArticle*, *ModifyArticle* und *SubmitFinalArticleVersion* zum *Article* Feature, sind *Reportlist* und *DelegateArticle* den Features *Role* bzw. *Delegation* zuzuordnen.

Die nächste Ebene der Design-Struktur weist die Umsetzung der Sub-Features auf, die ebenfalls als Makro realisiert sind. Diese Ebene beschreibt die Geschäftslogik der Sub-Features. Abbildung 4.4 zeigt diese Ebene anhand des *SubmitArticle* Sub-Features des OCS auf, das die Einreichung von Konferenzbeiträgen modelliert. Im Wesentlichen wird der Kontrollfluss für den Benutzer durch die drei Interaktions-SIBs, zu erkennen an der auf sich selbst zeigenden Kante (Schleife), bestimmt. Der Einreichungsprozess kann in die folgenden Abschnitte unterteilt werden:

1. Das durch den SIB *PreShowArticleSubmitForm* instanziierte Einreichungsformular wird von *ShowSubmitArticle* in Form einer HTML-Seite dem Dienstbenutzer angezeigt. Sobald dieser den Request absendet, werden die ausgefüllten Pflichtfelder und die Artikeldatei zum WebServer hochgeladen.

2. Nachfolgende SIBs überprüfen die auf der Server-Seite eingehenden Daten und bereiten diese für die Bestätigung der selbigen durch den Benutzer auf.

3. *ShowConfirmArticle* zeigt die am Server eingegangenen Daten an und wartet auf die Bestätigung der Richtigkeit. Nach Freigabe der Daten durch den Dienstbenutzer wird der Kontrollfluss weiter abgearbeitet.

4. Der weitere Workflow generiert Objekte, füllt diese mit Daten und leitet diese an die Persistenzschicht, z.B. Datenbank, weiter. Die Konferenzleiter werden über die Einreichung eines neuen Beitrages und der Dienst-

benutzer (Autor) über die erfolgreiche bzw. fehlerhafte Einreichung per Email informiert.

5. *ShowSubmitArticleAcknowledgement* zeigt abschliessend dem Einreicher des Beitrages eine Bestätigungsseite an, die nochmals die Daten zur Einreichung anzeigt.

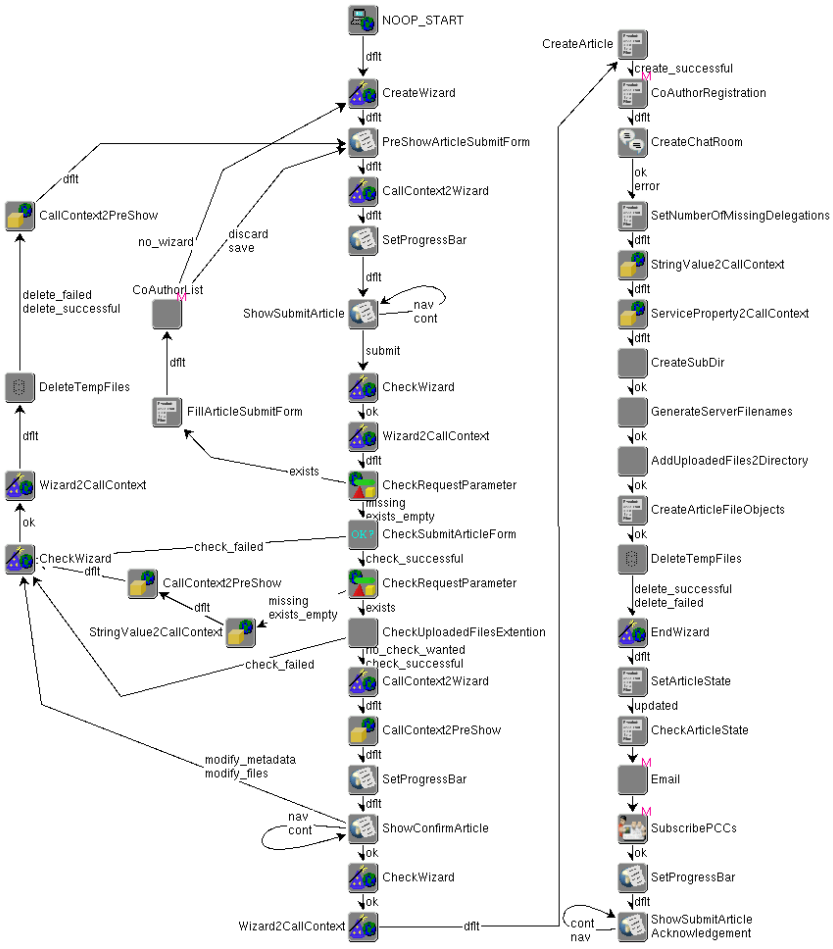


Abbildung 4.4: Struktur eines Sub-Features: Das Submit Article Feature

Die vorgestellte feingliedrige Strukturierung von Features stellt gleichzeitig die Basis für ein personalisiertes Rollen/Rechtesystem dar, das die Zugriffe auf die einzelnen Features steuert. So kann das betrachtete Feature *Submit Article* explizit der Rolle des Autors zugewiesen werden, um Inhabern dieser Rolle das Recht zu verleihen, einen Beitrag (Artikel) einzureichen. Die Untergliederung von Features ist darauf ausgelegt, verschiedene Anforderungen in verschiedenen Kontexten zu erfüllen. Sub-Features und ihre Granularität

werden bestimmt durch Konfigurationsanforderungen und dem Aufbau des Rollen/Rechtesystems. Im OCS können mit diesem Konzept Zugriffe auf sensible Daten und Funktionalitäten des Dienstes gezielt vergeben bzw. entzogen werden. Die genaue Vorgehensweise wird in Kapitel 4.2 beschrieben.

4.2 Rollen/Rechte Management

Ein kontrollierter Zugriff auf die Dienstfunktionalitäten und die vom Dienst verwalteten Daten ist *ein*, wenn nicht der wichtigste Punkt für die Entwicklung von sicheren Applikationen. Sensible Daten müssen jederzeit vor unautorisierten Zugriffen geschützt werden. Hierzu wurde ein Rollen-basiertes Rechtemanagement entwickelt, das für den OCS konzipiert wurde und in anderen, seiner Dienstfamilie angehörigen, Applikationen ebenfalls erfolgreich eingesetzt wird. Dieses Management zeichnet sich durch seine hohe Flexibilität aus, das für dynamische Rollen und Rechteabhängigkeiten realisiert wurde, wie es z.B. im OCS der Fall ist. Dessen Rechtezuordnungen stellen eine höhere Dynamik als in anderen Standard RBAC [4] [FK92]-Anwendungen dar. Eine Rolle besteht aus einer Gruppierung von Rechten, die dem jeweiligen Benutzer dieser Rolle die spezifischen Zugriffe und Aktionen auf Funktionalitäten des Dienstes oder der Objekte erlauben.

Das Rollen/Rechte Management ist, wie die meisten Funktionalitäten der Applikation, ebenfalls als ein Feature realisiert. Im Hauptgraph (siehe Abbildung 4.2) des OCS ist dieses Feature als Makro mit der Benennung *Roles* wiederzufinden. Die Abbildung 4.5 zeigt den SLG des Makros, auf dessen Funktionalitäten nachstehend näher eingegangen wird.

Der Kontrollflussgraph ist in die drei Bereiche **Create Role**, **Modify Role** und **Delete Role** unterteilt, die, ausgehend von dem Interaktions-SIB *ShowRoleRightsMatrix*, erreichbar sind. Dem Benutzer des Features *Roles* wird mittels des SIBs *ShowRoleRightsMatrix* eine HTML-Seite angezeigt, dessen Inhalt die Auflistung aller im Dienst existierenden Rollen ist. Diese Auflistung enthält sowohl vordefinierte Rollen als auch Rollen die zur Laufzeit der Applikation online angelegt wurden. Der Zugriffskontrollmechanismus des OCS basiert auf dem Rollen/Rechtesystem und steuert die Zugriffe auf die Features und Sub-Features des Dienstes. Das Rollen/Rechtesystem ist selbst als ein Feature entwickelt und wird ebenfalls über den Zugriffskontrollmechanismus verwaltet.

Der in Abbildung 4.5 mit **Create Role** gekennzeichnete Teilgraph spiegelt den Kontrollfluss für die Erstellung einer neuen Rolle wider. Der Interaktions-

[4] Role Based Access Control

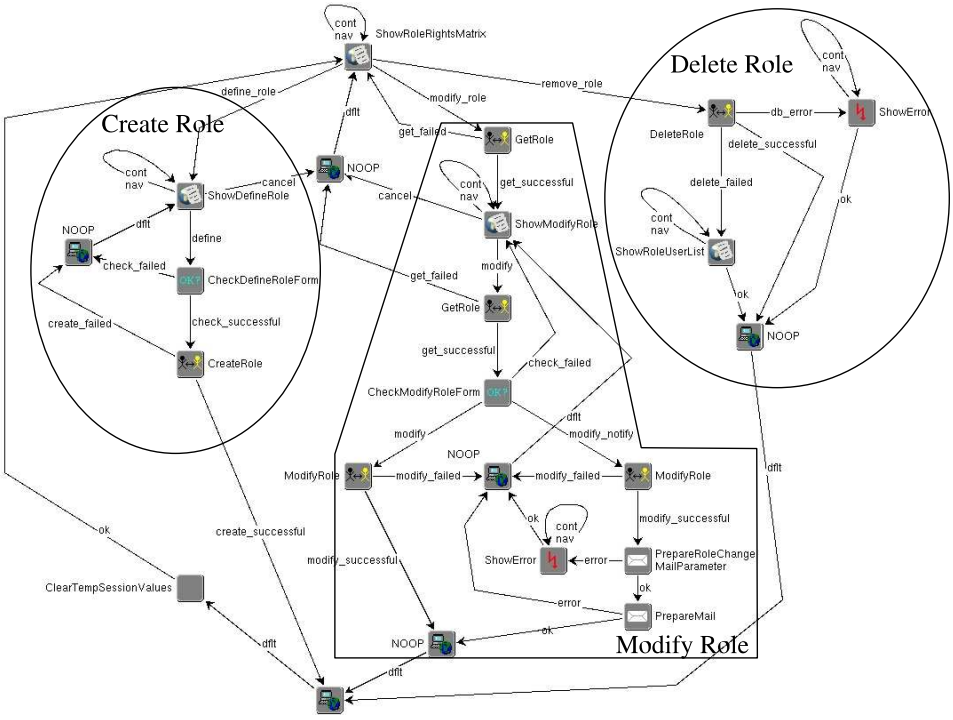


Abbildung 4.5: Role Management Service im OCS: Das Graph-Modell

SIB *ShowDefineRole* zeigt ein Formular an, mittels dessen der Name und eine kurze Beschreibung der neu anzulegenden Rolle anzugeben ist. Des Weiteren wird eine Liste von auswählbaren Rechten angeboten. Die der Rolle zugewiesenen Rechte legen den Zugriff auf die Features bzw. der Sub-Features fest und haben einen direkten Einfluss auf die Benutzersicht bzgl. des Dienstes. Nach Betätigung des *Save* Buttons werden die Daten übertragen und der Aktions-SIB *CheckDefineRoleForm* ausgeführt, der die Definition der neuen Rolle auf Abhängigkeiten überprüft und Fehler aufzeigt. Ist die Definition der Rolle wohlgeformt, wird diese von *CreateRole* als ein neuer Eintrag in der Role-Tabelle der Persistenzschicht abgelegt. Der Prozess wird, bei erfolgreicher Erstellung der neuen Rolle, mit der Anzeige der Liste aller Rollen inklusive aller Neuen angezeigt.

Der Teildienst **Modify Role** wird über die Kante *modify_role* erreicht, wenn ein Benutzer den *Modify Role* Button einer Rolle benutzt. Der nächste auszuführende SIB *GetRole*, der ein Rollen-Objekt instanziiert und die entsprechenden Daten aus der Datenbank liest, reicht die Rolle an *ShowModifyRole* weiter, der diese editiert. Nach Modifikation der Rolle und Absenden des Requests wird die Rolle eingelesen und die Modifikationen mittels *CheckMo-*

difyRoleForm überprüft. Im Fehlerfall wird die veränderte Rolle unter Angabe der aufgetretenen Fehler zur Korrektur erneut angezeigt. Im Falle der erfolgreichen Validierung der Änderungen wird das Role-Objekt in der Datenbank gespeichert. Benutzer, die zu diesem Zeitpunkt mit der alten Rollendefinition arbeiten, behalten diese zum Schutz vor Inkonsistenzen im Kontrollflussverhalten bis zur Abmeldung bei. Erst für eine erneute Anmeldung wird die neue Rollendefinition wirksam. Zusätzlich können die Inhaber der modifizierten Rollen über die Änderung der Rolle per Email informiert werden. Abschließend wird automatisch zur Rollenübersicht gegangen.

Der dritte Teildienst **Delete Role** ist für die Löschung von ausgewählten Rollen zuständig. Für die Löschung von Rollen gelten die folgenden Randbedingungen:

1. Vordefinierte Basis-Benutzerrollen, d.h. die mit dem Dienst ausgelieferten Rollen dürfen nicht gelöscht werden.
2. Es dürfen nur neu angelegte Benutzerrollen gelöscht werden, die keinem Benutzer zugewiesen sind. Betroffene Benutzer werden sonst aufgelistet und der Löschvorgang abgebrochen.

Das Rollen/Rechtesystem ist sehr flexibel, da es sich zur Laufzeit um neue Rollen erweitern lässt, bestehende Rollen können rekonfiguriert werden und die Zuordnungen der Rollen zu Dienstbenutzern sind frei wählbar.

4.2.1 Das Benutzer/Rollen-Rechtemodell

Nach der Vorstellung der Strukturierung der Features und des Rollen/Rechtesystems auf Graphebene, stellt dieses Kapitel die beiden Themen in Beziehung. Die mit den Features bzw. Sub-Features modellierten Funktionalitäten des Dienstes spezifizieren Dienstkomponenten, die einem Dienstbenutzer zur Verwendung freigeschaltet werden können. Das Benutzer/Rollen-Rechtemodell (siehe Abbildung 4.6), stellt den Zusammenhang zwischen Benutzern, Rollen und den Features (Dienstkomponenten) her. Auf diese Modell wird im Folgenden näher eingegangen.

Definition 4.2.1 legt die Relation zwischen Benutzern und Rollen fest und beschreibt die Struktur der Rollen untereinander.

Definition 4.2.1 (Benutzer/Rollen Relation)

- *Ein Benutzer ist mindestens einer Rolle zugeordnet.*

- *Eine Rolle kann mehreren Benutzern zugewiesen sein.*
- *Es ist immer nur eine Rolle zu einem Zeitpunkt für eine Benutzerinstanz aktiv.*
- *Rollen sind unabhängig voneinander.*

Die genauen Rollen/Rechte Abhängigkeiten werden in der Definition 4.2.2 aufgezeigt.

Definition 4.2.2 (Rollen/Rechte Relation)

- *Eine Rolle ist eine Ansammlung von Rechten.*
- *Ein Recht gibt die Erlaubnis für den Zugriff auf ein Objekt oder ein Feature.*
- *Ein Recht ist eindeutig, d.h. es existiert kein weiteres Recht mit den selben Eigenschaften.*
- *Rechte werden in Feature-Kategorien gruppiert.*
- *Eine Rolle kann Rechte verschiedener Feature-Kategorien enthalten.*

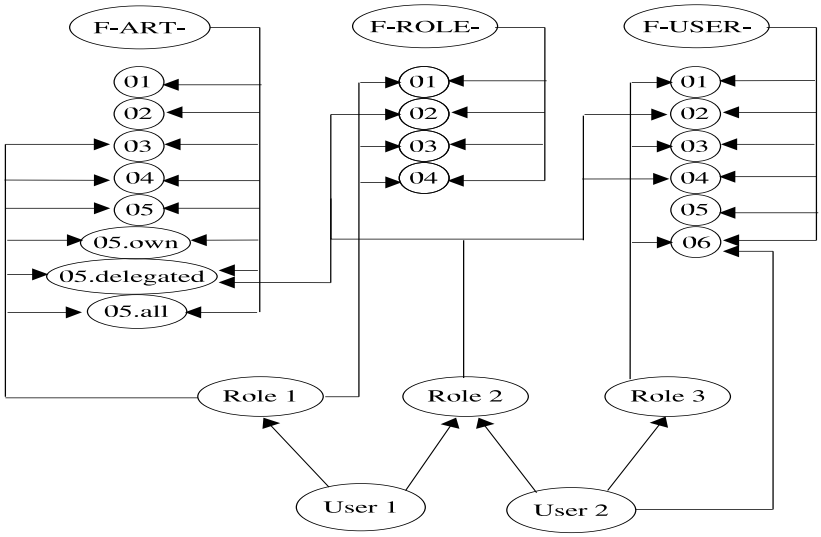


Abbildung 4.6: Das Benutzer/Rollen/Rechte Modell

Beschreibung der Rechte	**Synonym**
Zugriff auf Artikellisten: eigene, zugewiesene oder alle Artikel	F-ART-06
Speicherung von eigenen Notizen	F-ART-07
Modifikation der Artikel	F-ART-10
Löschung der Artikel	F-ART-12
Einreichung einer Zusammenfassung eines Artikels (abstract)	F-ART-16
Modifikation der Zusammenfassung (abstract)	F-ART-17
Einreichung der Artikelversion zur Zusammenfassung	F-ART-19
Verwaltung der hochgeladenen Artikeldateien	F-ART-21
Delegierung eines Begutachtungsauftrages	F-DEL-01
Annahme oder Ablehnung eines Begutachtungsauftrages	F-DEL-02
Erstellung und Sendung von Emails	F-MAIL-05

Tabelle 4.1: Auszug einiger OCS Features

Die Abbildung 4.6 stellt die Zusammenhänge grafisch dar. Die aufgezeigten Benennungen **F-ART** oder **F-ROLE** sind Feature-Kategorien, die Features gruppieren, welche Dienstfunktionalitäten entsprechen. Das Benennungsschema basiert auf eindeutigen Namen und ist wie folgt aufgebaut:

```
F-<FeatureKategorie>-<SubFeatureID>.<Filter>
```

- Die **FeatureKategorie** bezeichnet einen Hauptdienst der Applikation, der als eigener SLG realisiert ist.
- Die **SubFeatureID** referenziert einen dem Hauptdienst zugeordneten Dienst, der als eigener SLG oder SIB umgesetzt wurde.
- Das Suffix **Filter** ist optional und dient zur feingranularen Spezifizierung einer Berechtigung.

Der beschriebene Aufbau lässt sich einfach an der Feature-Kategorie **Article** verdeutlichen, welche mit dem Synonym **ART** bezeichnet wird. Die einzelnen Sub-Features einer Feature-Kategorie erhalten eindeutige Namen, indem dem Haupt-Feature eine fortlaufende Nummerierung angehängt wird (siehe Abbildung 4.6). Zum Beispiel verleiht das Feature *F-ART-03* das Recht, neue Beiträge einzureichen. Zur Laufzeit wird überprüft, ob der Dienstbenutzer dieses Recht hat, bevor der Einreichungsmechanismus zur Verfügung gestellt wird. Die Tabelle 4.1 zeigt einen Ausschnitt weiterer Rechte.

Ein Beispiel für die Verwendung von **Filter** ist die Erlaubnis, Artikel lesen zu dürfen, *F-ART-05*. Die Erlaubnis sagt nur aus, dass der Teildienst, der einen Artikel zum Lesen bereit stellt, verwendet werden darf, aber nicht welche der Artikel anzuzeigen sind. Hier wird mittels des **Filters** unterschieden, ob nur die eigenen Artikel (*F-ART-05.own*), nur die delegierten Artikel (*F-ART-05.delegated*) oder alle Artikel (*F-ART-05.all*) gelesen werden dürfen.

4.2.2 Zugriffsrechte zur Laufzeit

Zur Laufzeit der Applikation sind alle Features, die auf der Modellierungsebene definiert worden sind, existent. Die Inhalte einer Seite und somit auch die Funktionalitäten des Dienstes sind von der Rolle, mit der sich ein Benutzer in den Dienst angemeldet hat, abhängig. Die Rolle beschreibt die Dienstfunktionalitäten und die darauf erlaubten Aktionen, die ein Benutzer sehen bzw. ausführen darf. Im Folgenden wird detaillierter auf die Bestimmung der Zugriffsrechte zur Laufzeit eingegangen. Die Darstellung wird am Beispiel des OCS vorgenommen.

Nach dem erfolgreichen Anmeldevorgang wird der Benutzer in den internen Bereich der Anwendung geleitet. Aufgrund seiner Rolle wird die Sicht auf den Dienst in dem Sinne personalisiert, dass die Navigationsleiste die dem Benutzer zugeordneten Rollen und die ihm erlaubten Feature-Links beinhaltet. Jeder Feature-Eintrag in der Navigationleiste führt zu einem Hauptfeature, wobei die Sub-Features selbst bzw. deren Verlinkung in der Informationsseite zu finden sind. Dieses Vorgehen lehnt sich an die in Kapitel 4.1.2 beschriebenen Feature-Struktur an.

Die Zugriffsrechte zur Laufzeit hängen von mehreren Aspekten ab. Es können z.B. Einreichungen gruppiert werden, um Benutzern gezielt Zugriff auf Artikelkategorien zu gewähren. Konflikte können zwischen einem Benutzer und Artikel spezifiziert werden und stellen einen eingeschränkten Zugriff auf die Artikel dar. Diese Aspekte entsprechen einer Filterung der im Dienst vorhandenen Artikel. Der Ablauf von Konferenzen ist in Phasen unterteilt und unterliegt Zeitvorgaben, Rollen vergeben Berechtigungen, und Zugriffe auf Objekte sind zustandsbehaftet. Die Zugriffsrechte auf Objekte im Dienst sind somit dynamisch und können sich jederzeit ändern. Für den OCS lassen sich die folgenden Punkte festhalten, die die Zugriffsrechte bestimmen:

- Die Rechte der Rolle des Benutzers (user).
- Die Zugriffsrechte, die abhängig von dem Status des Objektes sind.

- Die Konferenzrestriktionen, die z.B. von den Konferenzzeitfristen abhängen. Nach Ablauf einer Zeitvorgabe, z.B. der Zeitraum in dem die Artikeleinreichung möglich war, werden die entsprechenden Rechte zur Verwendung des Features entzogen.

Diese Art von Rechten bestimmen die Zugriffe bzw. Aktionen auf Objekte. Die Menge der Rechte sind zu Beginn einer Konferenz festgelegt und somit statisch zur Laufzeit des Dienstes. Die im Dienst verwalteten Objekte sind zustandsbehaftet, die in Abhängigkeit ihres jeweiligen Status verschiedene Zugriffsrechte voraussetzen. Die Status-basierten Rechte der Objekte sind zur Laufzeit der Applikation festgelegt. Die Zuordnung zwischen Rollen und Benutzern und die Zuweisung von Rechten zu Rollen sind zur Laufzeit konfigurierbar. Dieses, unter Berücksichtigung der Möglichkeit, neue Rollen zu erstellen, macht das Konzept sehr flexibel und im Hinblick auf die Rechtevergabe sehr dynamisch. Dieses projiziert sich ebenfalls auf die Benutzersicht. Funktionalitäten werden entweder automatisch durch den Dienst (z.B. Übergang zur nächsten Phase) oder manuell durch den Konferenzleiter ein- bzw. ausgeschaltet. Durch die Anzahl der vorgestellten Aspekte und der daraus folgenden Dynamik der Rechte, wird die Autorisierung für einen Zugriff für jede Anfrage neu berechnet und überprüft:

1. Berechnung der Benutzerrechte *Perm* (Permissions) unter Berücksichtigung der Rollenrechte, der Objektrechte und der Konferenzrestriktionen:

$$Perm = (role_{perm} \cap Object_{perm}) \setminus conference_{restr}$$

2. Überprüfung, ob die berechneten Rechte die nötigen Zugriffsrechte für das entsprechende Objekt haben:

$$Access_{perm} \subseteq Perm$$

3. Bei erfolgreicher Prüfung wird dem Benutzer erlaubt, die durch das Recht spezifizierte Aktion auf dem Objekt durchzuführen.

Beispiel: Lesezugriff auf einen Artikel

Das Beispiel beinhaltet die Berechnung des Zugriffsrechtes für das Feature F-ART-05, das einem Benutzer das Lesen von Artikeln erlaubt. Zu berücksichtigen sind die Rechte des Benutzers, die Restriktionen der Konferenz und die benötigten Rechte des Objektes. Die Benutzerrechte werden nach der Anmeldung aus der Datenbank gelesen und bleiben für die Session unverändert.

Die Konferenzrestriktionen und die Objektzugriffsrechte werden in Dateien definiert.

Ein Konferenzstatus ist gültig zwischen zwei Zeitfristen (deadlines). Jeder Konferenzstatus spezifiziert Restriktionen, die Dienstfunktionalitäten bis zum Ablauf der Zeitfrist verbieten. Das Entscheidungssystem verfügt über mehrere Deadlines, die gleichzeitig die Phasen des Entscheidungssystems repräsentieren. Als Beispiel wird die Definition der *single report deadline* gezeigt, die Features auflistet, die bis zum Ablauf der Deadline verboten sind:

SingleReportDeadline =
F-ART-03, F-ART-16, F-ART-17, F-ART-19

Der Artikelstatus basiert auf einer Zustandsmaschine, die in Abhängigkeit eines Ereignisses in den nächsten Zustand, wie *created*, *abstract*, *submitted* oder *delegated* übergeht (Transition). Die erlaubten Berechtigungen im Status *Delegated* (der Artikel wurde zur Begutachtung weitergeleitet), sind wie folgt definiert:

Delegated =
F-ART-05, F-ART-06, F-ART-07,
F-ART-10, F-ART-12, F-ART-21, F-BID-01,
F-DEL-01, F-DEL-02, F-MAIL-05

Es wird davon ausgegangen, dass die *SingleReportDeadline* die nächste ablaufende Deadline ist, der Status des Artikels *Delegated* aufweist, den ein Benutzer beabsichtigt zu lesen, und dass die Rolle des Benutzers das Leserecht *F-ART-05* erlaubt. Wird die *SingleReportDeadline* und der *Artikelstatus* zusammen mit dem Recht des Benutzers, den Artikel laut seiner Rolle lesen zu dürfen betrachtet, so wird dem Benutzer der Artikel zum Lesen freigegeben. Das ist daran zu erkennen, dass der Konferenzstatus das Lese-Feature nicht verbietet und der Artikelstatus *Delegated F-ART-05* erlaubt.

4.2.3 Personalisiertes Rechtemanagement

Der in diesem Kapitel betrachtete Ansatz ist eine Erweiterung des Benutzer/ Rollen-Rechtemodells, das in Kapitel 4.2.1 vorgestellt wurde. Das existierende Modell weist eine Flexibilität der Rechtevergabe in Bezug auf Rollen auf. Die Bündelung von Rechten mittels Rollen, die zur Laufzeit modifiziert und erstellt werden können ist ein adäquates Mittel zur Verwaltung und Zuweisung von Rechten für Konferenzen. In Konferenzen sind die Rechte der Benutzer einer Rolle über den Begutachtungsprozess annähernd gleich. Die Erfahrung hat gezeigt, dass in manchen Situationen eine noch höhere Flexibilität von

Nöten ist, um alle Anforderungen komfortabel erfüllen zu können. Dieses wurde ersichtlich beim *Online Journal Service* (OJS), der Web-Applikation für die Unterstützung von Herausgebern wissenschaftlicher Zeitschriften. Der Dienst gehört zur Dienstfamilie des OCS und basiert auf gemeinsamen Funktionalitäten wie *Article*, *Delegation* und dem *Report* Feature. Unterschiede treten im Entscheidungsprozess und in der Laufzeit auf. Der Entscheidungsprozess des OJS weist einen asynchronen Begutachtungsprozess in der Hinsicht auf, dass die Einreichungen einen eigenen Prozess durchlaufen, anders als bei der synchronen Evaluation beim OCS, in der alle Beiträge konkurrierend involviert sind. Für die Entscheidungsfindung wird ein zum OCS verfeinertes Prozessmanagement eingesetzt, das eine beliebige Anzahl von Begutachtungsdurchgängen von überarbeiteten Artikeln (Revisionen) verwaltet. Dieser Workflow bringt eine größere Anzahl von leitenden Rollen mit sich (Editor, Editor in Chief, Guest Editor, Editorial Office, ...). Aufgrund der viel längeren Laufzeit eines Journals im Vergleich zu den drei bis vier Monaten einer Konferenz kommt es gewöhnlicherweise zu Änderungen der persönlichen bzw. beruflichen Situation einzelner Benutzer, was Auswirkungen auf ihre Rechte im laufenden Dienst hat. Dieses verlangt nach einer individuellen Anpassung der erlaubten Sichten und Aktionen des Benutzers an den Dienst mittels eines feingranularen Managements für Rollen und Rechte.

Das Benutzer/Rollen-Rechtemodell wurde um eine weitere Schicht erweitert, die Benutzerrechte verwaltet. Individuelle Rechte von Einzelnen konnten somit hinzugefügt bzw. entzogen werden, ohne die gesamte Rolle zu modifizieren. Die im Folgenden detaillierter beschriebene Erweiterung wurde als Feature realisiert und konnte so auf einfache Weise ebenfalls in den OCS integriert werden und erwies sich dort als sehr nützlich. Ein Beispiel ist der Ablauf der Einreichungsfrist für Artikel. Soll z.B. einigen Autoren gezielt das Einreichungsrecht F-ART-03 nach Ablauf der Deadline verliehen werden, damit diese weiterhin Beiträge hochladen können. Dieses war zuvor nur möglich, indem der gesamten Rolle und somit allen Autoren das Recht eingeräumt würde.

Personalisierung: Benutzer/Rechte-Management

Die Personalisierung mittels Benutzerrechten ist als erweiterndes Konzept zum bestehenden Benutzer/Rollen-Management entstanden. Das Konzept vereinigt die globale Definition von Rechten über Rollen mit der Flexibilität des individuellen Rechtemanagements eines Benutzers. Die Benutzer/Rollen Definition lautet:

Definition 4.2.3 (Benutzer/Rechte Relation)

- *Ein Recht kann einem oder mehreren Benutzern erteilt werden.*
- *Ein Recht kann explizit einem Benutzer entzogen werden, d.h. dieses wird als Restriktion vermerkt.*
- *Ein Benutzer kann mehreren Rechten aus verschiedenen Feature-Kategorien zugeordnet sein.*

Die Zuordnung bzw. der Entzug eines Rechtes geschieht immer mit Bezug auf eine Rolle, sodass die individuelle Vergabe der Rechte für jede Rolle verfeinert werden kann. Diese Vorgehensweise stellt eine erhöhte Flexibilität der Einstellungen dar. Unter Berücksichtigung der neuen Relation hat sich die Berechnung des Rechtes für einen Zugriff oder die Benutzung einer Dienstfunktionalität wie folgt geändert:

$$Perm = ((role_{perm} \cup user_{perm}) \cap Object_{perm}) \setminus user_{restr} \setminus conference_{restr}$$

$$Access_{perm} \subseteq Perm$$

Die in Kapitel 4.2.2 eingeführte Formel ist um die Permissions und Restriktionen des Benutzers (user) erweitert worden. Realisiert ist die Erweiterung, wie alle Dienste in der OCS-Familie als ein eigener SLG, der dem Feature *Users* zugeordnet ist. Der SLG des personalisierten Rechtemanagements wird in Abbildung 4.7 gezeigt. In dem Graphen ist sehr gut die Verwendung des Wizardbasierten Datenaustausches zwischen den verschiedenen Kontexten (siehe Kapitel 3.2) zu erkennen. Objekte werden in der Dienstfamilie in einem Kontext gehalten, der der jeweils eindeutigen Anfrage (Request) zugeordnet ist und dadurch die Threadsicherheit gewährleistet. Der Benutzer kann parallel mit mehreren Browser-Fenstern auf denselben Workflow zugreifen.

Der Dienst besteht aus vier Teilfunktionalitäten:

- Der mit **user's roles** gekennzeichnete Teilgraph zeigt eine Liste von Rollen (*ShowUserRoleList*), die einem Benutzer zugewiesen sind. Die Information über die Rollenzuordnung wird aus der Datenbank mittels des SIBS *UserRoleList* gelesen und dem Wizard mit dem SIB *Context2Wizard* übergeben, der Objekte Kontext-übergreifend im Cache hält. *Context2PreShow* schreibt die Objekte in einen separaten Kontext, der alle benötigten Objekte für das Füllen der HTML-Seite beinhaltet.

- Der Teilgraph **user's permissions** stellt den Kontrollfluss von der Auswahl der Rolle, für die Benutzerrechte spezifiziert werden sollen, bis hin zur Anzeige der Seite durch *ShowUserPermissions* dar. Der SIB *UsersPermissionList* liest die Benutzerrechte und zeigt die Benutzerspezifischen Rechte und Restriktionen mit den Rechten der ausgewählten Rolle an.

- Die Berechtigungen und Restriktionen können nun modifiziert werden. Nach dem Absenden des Formulars werden die Daten Server-seitig überprüft und mittels des SIBs *ShowConfirm* zur Bestätigung angezeigt. Der Workflow **confirmation** wird mit der Bestätigung abgeschlossen.

- Zum Abschluss werden alle Berechtigungen und Restriktionen in der Datenbank gespeichert, dieses wird von **stores permissions** durchgeführt.

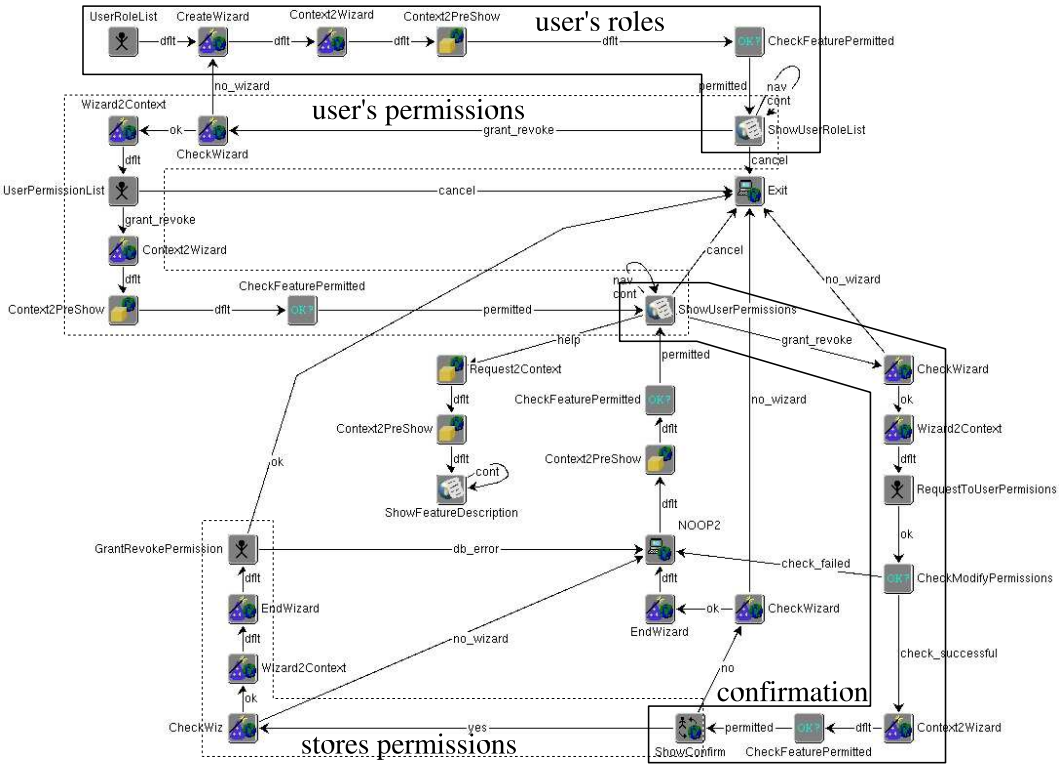


Abbildung 4.7: Benutzer/Rollen-Management: SLG

Der Zugriff auf ein Feature wird graphbasiert mit dem SIB *CheckFeaturePermitted* modelliert und zur Laufzeit überprüft. Die aktuellen Rechte des Benutzers werden zusätzlich dynamisch mittels der expliziten Benutzerberechtigun-

gen gefiltert. Der SIB *CheckFeaturePermitted* vergleicht vor einem Zugriff die nach der Filterung übergebliebene Menge von Rechten mit den erforderlichen Features des Objektes.

Es wurde ein hohes Maß an Flexibilität und Einfachheit bei der Konfiguration von Berechtigungen mit einer minimalen Erhöhung des Prüfvorgangs zur Laufzeit erzielt. In der Praxis konnte so einem *Editor* im OJS individuell das Recht für die benötigte Sicht auf die Begutachtungsverteilung gegeben werden, ohne der Rolle *Editor* und somit allen Rolleninhabern diese Sicht zu geben. Es ist einfacher und komfortabler auf Basis einer allgemein definierten Rolle, Rechte an Benutzer zu verleihen und gezielt einem Benutzer das Recht zu entziehen, bestimmte Informationen über seinen Artikel zu sehen, ohne andere Dienstbenutzer zu beeinflussen. Es ist weiterhin möglich, zentralisiert Rechte über eine Rolle an die Inhaber zu vergeben bzw. zu entziehen. Ein Ansatz die Rechtevergabe gänzlich über direkte Benutzer/Rechte-Relationen zu realisieren, wäre mit einem extremen Wartungsaufwand verbunden, da jede Änderung einer Rolle dann für jeden betroffenen Benutzer durchgeführt werden müsste.

4.2.4 Validierung des Zugriffskontrollmechanismus

Wie das zugrunde liegende Rollen/Rechte-Management, spielt die Erweiterung des Zugriffsmechanismus eine ebenso wichtige und zentrale Rolle im Umgang mit vertrauensvollen Daten und den hierzu konzipierten Dienstfunktionalitäten. Umso größer ist die Bedeutung des Wissens, den Kontrollflussgraphen als sicher in Bezug auf die Erfüllung von Bedingungen (constraints) modelliert zu haben. Hierzu wird der Benutzer/Rechte-Graph ebenfalls einer Model Checking Validierung unterzogen und kann bei erneuten Änderungen schnell auf seine Richtigkeit hin überprüft werden.

Der in Abbildung 4.7 dargestellte und bereits erörterte Kontrollflussgraph des Benutzer/Rechte-Managements, weist die drei Interaktions-SIBs *ShowUserList*, *ShowListPermissions* und *ShowConfirm* auf. Diese SIBs zeigen HTML-Seiten mit vertrauenswürdigen Rechtezuweisungen an. Der Inhalt dieser Seiten und die inkludierte Möglichkeit zu Modifikationen der Rechtezuweisungen muß vor unautorisierten Zugriffen geschützt werden. Es ist zu überprüfen, ob der zugreifende Benutzer die benötigten Rechte für das entsprechende Feature und die damit zusammenhängenden Informationen besitzt. Die Überprüfung der Zugriffsvoraussetzungen werden im Graphmodell mit dem SIB *CheckFeaturePermitted* vor jedem Interaktions-SIB modelliert und zur Laufzeit berücksichtigt.

Für das Model Checking werden die drei SIBs, die jeweils eine HTML-Seite

zur Interaktion mit dem Dienstbenutzer anzeigen, zur Gruppe des *ShowSIB* gezählt. Für die Validierung werden alle *ShowSIB* mit der atomaren Proposition `ShowSIB`s im SLG annotiert.

Der nächste Schritt ist die Definition der temporal logischen Formel, die die Eigenschaften beschreibt, dass direkt vor jedem *ShowSIB* der SIB *CheckFeaturePermitted* vorkommen muss und die verbindende Kante `permitted` lautet (siehe Abb. 4.7).

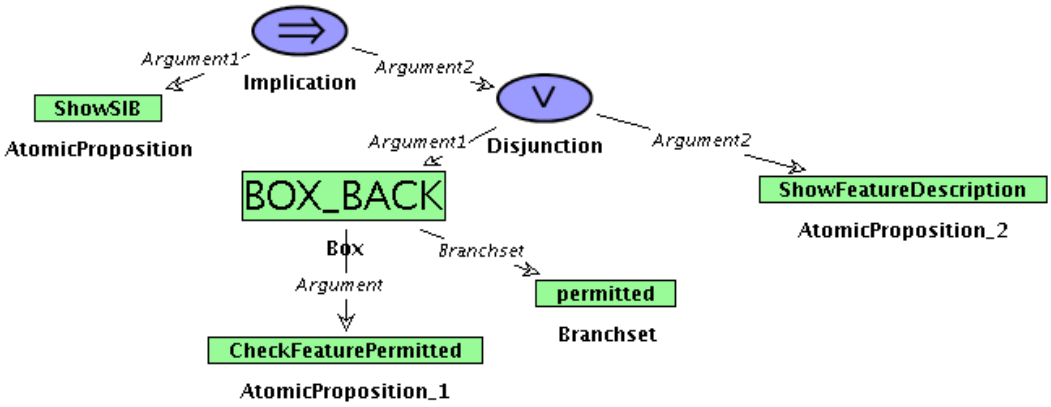


Abbildung 4.8: Benutzer/Rollen-Management: Temporal logische Formel

Die temporallogische Formel lässt sich ihrerseits mittels des FormulaBuilders [JMS06] als Graph modellieren (siehe Abbildung 4.8) und zu CTL Syntax übersetzen:

`'ShowSIB => ![permitted]! CheckFeaturePermitted` $\vee$
`'ShowFeatureDescription`

In der Formel ist der SIB *ShowFeatureDescription* von der eigentlichen Sicherstellung von unautorisierten Zugriffen ausgeschlossen worden, da er in diesem Zusammenhang als unkritisch angesehen wird.

Mittels des Model Checkers GEAR [BMRS06] wird diese temporallogische Formel gegen den Kontrollflussgraphen geprüft. In Abbildung 4.9 wird das Ergebnis der Validierung grafisch dargestellt. In diesem fehlerhaften SLG ist der Pfad vom SIB *UserPermissionList* über die Kante *permitted* zum SIB *ShowUserPermissions* als Fehlerpfad hervorgehoben. Dies begründet sich darin, dass laut definierter Formel nur der SIB *CheckFeaturePermission* mit *ShowUserPermissions* über die Kante *permitted* verbunden sein darf, um die benötigte Zugriffsüberprüfung sicherzustellen.

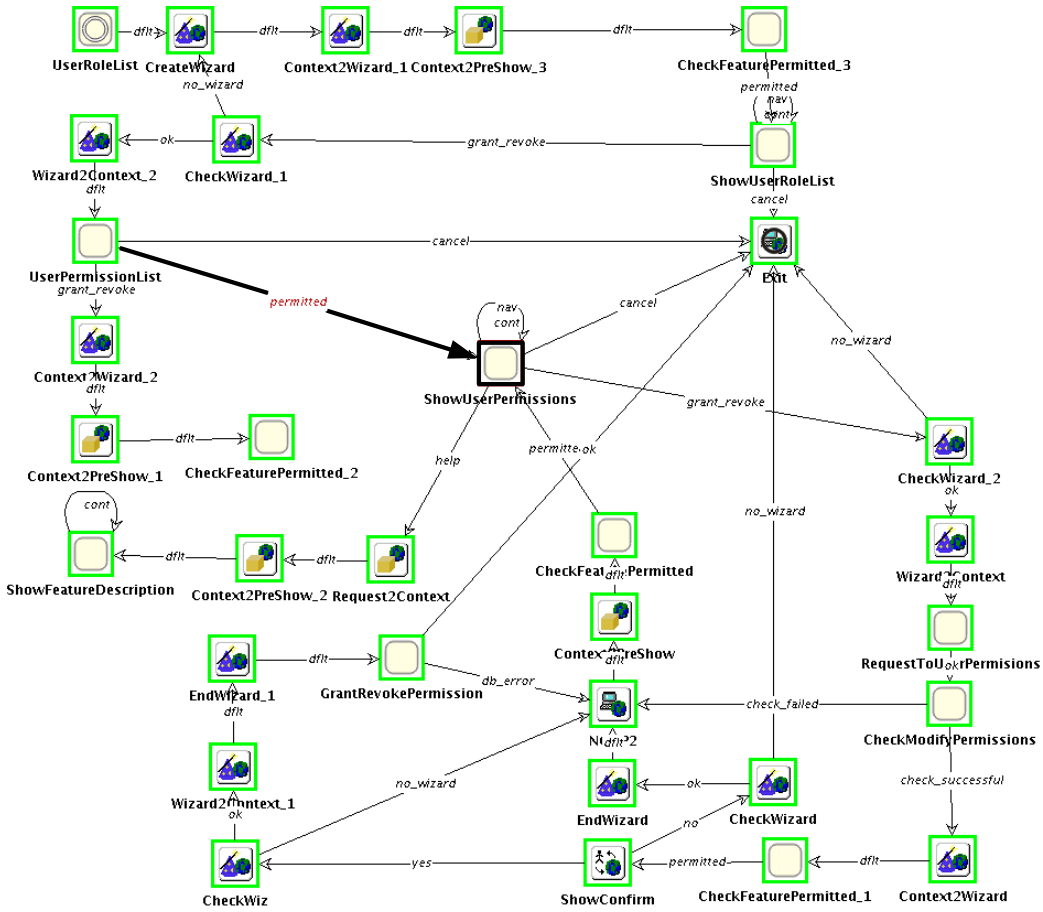


Abbildung 4.9: Benutzer/Rollen-Management: Validierung

4.3 Bidding- und Verteilungsmechanismus

Das Feature *Bidding* zählt mit zu den wichtigsten Funktionalitäten des OCS. In der Phase der Verteilungen der Begutachtungsaufgaben minimiert dieses Feature den Arbeits- und Koordinationsaufwand der Konferenzleiter drastisch. Wird die mit dem OCS durchgeführte Konferenz *Advanced Data Mining And Applications 2009* (ADMA) betrachtet, so zeigt sich mit 322 Artikeln, 1169 Verteilungen und 795 Benutzern die Größe der Konferenz. Eine adäquate Verteilung von Begutachtungsaufgaben in diesen Größenordnungen stellt die Konferenzleiter vor eine sehr große Herausforderung, die manuell nur mit einem immensen Zeit- und Arbeitsaufwand zu bewältigen wäre. Der Bidding- und Verteilungsmechanismus wurde realisiert, um diesen Aufwand so gering wie möglich zu halten und gleichzeitig die beste, auf die Konferenz abgestimmte,

Verteilung zu gewährleisten.

Eine gute Verteilung sollte die folgenden Punkte beachten:

- Alle Gutachter haben dieselbe Arbeitsauslastung.
- Artikel werden von Experten begutachtet.
- Mindestens drei Gutachter beurteilen einen Artikel.

Der Zeitrahmen für die Erstellung der Gutachten wird in der Regel von einer gemeinsamen Deadline bestimmt. Ein ausgeglichener Arbeitsaufwand stellt den Gutachtern das gleiche Zeitfenster für jeden Artikel zur Verfügung und gewährleistet jedem Artikel dieselbe zeitliche Aufmerksamkeit. Eine Ausnahme stellen Gutachter mit vielen SubReviewern dar, an die weiterdelegiert werden kann. Die Begutachtung von Artikeln durch Experten bzw. Gutachtern mit Wissen in dem Themenbereich des jeweiligen Beitrages, ist ein wichtiger Bestandteil für die gerechte Bewertung eines Artikels. Nicht außer Acht gelassen werden darf das Interesse eines Gutachters an einem Artikel. Zwangszuweisungen sind für den Begutachter nicht immer erfreulich und können u.U. die Beurteilung beeinflussen oder zu einer Ablehnung des Artikels führen. Letztendlich finden die Begutachtungen ehrenamtlich statt ohne jegliche Bezahlung. Um eine repräsentative Beurteilung und Entscheidungsfindung zu gewährleisten, werden Artikel von mindestens drei Gutachtern beurteilt.

Das *Bidding* Feature zeigt die Zuordnungen zwischen Artikeln und Gutachtern in einer Matrix an und unterstützt die Konferenzleiter in folgender Weise:

1. Die Gutachter stimmen für die zu begutachtenden Artikel:
   - **B (“bid”):** Sehr großes Interesse an der Begutachtung des Artikels.
   - **I (“indifferent”):** Geringes Interesse, aber bei einer Zuteilung würde die Begutachtung durchgeführt werden.
   - **U (“unable”):** Zu geringes Fachwissen, um eine sichere Beurteilung abgeben zu können.
   - **R (“rejected”):** Ablehnung der Begutachtung des Artikels.
   - **C (“conflict”):** Der Gutachter hat einen Konflikt mit den Autoren des Artikels. D.h. er kennt die Autoren und würde eventuell bei der Bewertung beeinflusst sein.
2. Berechnung der Arbeitsauslastung des einzelnen Gutachters auf Basis der mit **B** gewählten Artikel.

3. Berechnung einer Verteilung der Begutachtungsaufgaben mittels lp_solve [Ber09], ein externes Programm zur Lösung von linearen Optimierungproblemen.

4. Automatische Verteilung der Aufgaben an die Gutachter per Mausklick.

Diese Vorgehensweise wird dem Konferenzleiter empfohlen, damit der Aufwand minimal gehalten wird. Das Feature ist sehr flexibel, so dass Schritt 1. nicht stattfinden muss, um eine gleichmässige Verteilung mit Schritt 3. zu erzielen. Allerdings können Gutachter ohne Schritt 1. keinen direkten Einfluss auf die Verteilung nehmen und die Grundlage für das Delegieren an Fachkompetenzen des Themenbereiches des jeweiligen Artikels wäre nicht automatisch berücksichtigt. Die Konferenzleiter können die Delegierungen auch ohne die beschriebenen Schritte durchführen, indem diese manuell die Artikel/Gutachter-Beziehung setzen. Des Weiteren ist jederzeit eine manuelle Änderung der Verteilungsplanung und unter Berücksichtigung der dadurch auftretenden neuen Konstellationen erneute Berechnung der optimalen Lösung möglich.

ID	Title	Author	CoAuthors	Bidding	Type
001	OCS	Karusseit, Martin		(•) B ( ) I ( ) U ( ) C	Regular Paper
002	Bidding Feature	C1, PC		( ) B (•) I ( ) U ( ) C	Regular Paper
003	Feature-basiertes Design	Müller, Susi		( ) B ( ) I (•) U ( ) C	Regular Paper
004	Zugriffsrechte im OCS	Meier, Willi		( ) B ( ) I ( ) U (•) C	Regular Paper

Abbildung 4.10: Bidding Feature - Auswahlliste für Gutachter

Im Folgenden wird der Workflow des Bidding Features an einem kleinen Beispiel erläutert. Abbildung 4.10 zeigt die Liste, mittels derer die Gutachter für die einzelnen Artikel stimmen können. Die Wahl aller Gutachter wird in der Bidding-Matrix (siehe Abbildung 4.11) angezeigt. Die mit **B** gekennzeichneten Zellen der Matrix stellen die von den Gutachtern bevorzugten Artikel dar. Das Addieren der Gutachtenwünsche ergibt einerseits die Anzahl der Gutachten pro Artikel, andererseits die zu schreibenden Gutachten pro Gutachter, was deren Arbeitsauslastung entspricht. Abbildung 4.11 stellt die auf die Wünsche der Gutachter ausgerichtete momentane geplante Verteilungssituation dar. Die entsprechenden Checkboxen der Favoriten sind selektiert. Das betrachtete Beispiel setzt drei Gutachter pro Artikel voraus (siehe letzte Spalte `(3)`). Die Spalte `Sum` zeigt die momentane Anzahl pro Artikel an, wobei die Arbeitsauslastung in der letzten Spalte zu sehen ist.

ID	G1	G2	G3	G4	ID	Sum:(3)
001	B ☑	I ☐	C	I ☐	001	1
002	I ☐	I ☐	I ☐	U ☐	002	
003	U ☐	U ☐	B ☑	I ☐	003	1
004	C	B ☑	B ☑	I ☐	004	2
Sum:	1	1	2			

Abbildung 4.11: Bidding-Matrix - Anzeige der Benutzerwahl

Anders als in diesem vereinfachten Beispiel sind die Randbedingungen in der Praxis erheblich komplexer. In den bereits erwähnten Konferenzen TACAS und ADMA weist die Bidding-Matrix 3799 bzw. 39508 Zellen auf, was ohne einen Automatismus nicht mehr zu handhaben ist.

Zur Berechnung einer optimalen Lösung für die Verteilung von Artikeln an Gutachter bietet sich die lineare Optimierung an. Sie wird für die Bestimmung des Maximums bzw. Minimums einer linearen Funktion verwendet. Hierzu ist die Zielfunktion und die dazugehörigen Restriktionen zu erstellen, was letztendlich Eingaben für das Programm zur Bestimmung der Lösung sind. Im OCS ist die Zielfunktion wie folgt definiert:

$$\sum_{i}^{Artikel} \sum_{j}^{Gutachter} K_{i\,,\,j}\; X_{i\,,\,j} = MIN \tag{4.1}$$

Der Koeffizient $K_{i\,,\,j}$ wird bestimmt durch die im Dienst spezifizierten Konflikte zwischen Gutachtern und Artikeln, der Arbeitsauslastung des Gutachters und der Artikelauswahl des Gutachters. Die zu bestimmenden Variablen $X_{i\,,\,j}$ spezifizieren die Zuordnung eines Begutachtungsauftrages zwischen einem Artikel und einem Gutachter.

Die Restriktionen werden durch unterschiedliche Gleichungen, Ungleichungen definiert. Gewollte bzw. ungewollte Delegierungen zwischen Artikeln und Gutachtern werden durch die Gleichungen 4.2 festgelegt.

$$\begin{aligned} X_{Artikel\,,\,Gutachter} &= 1 \quad (Gesetzte\ Zuweisung) \\ X_{Artikel\,,\,Gutachter} &= 0 \quad (Verbotene\ Zuweisung) \end{aligned} \tag{4.2}$$

Für zu bestimmende Delegierungen wird der Wertebereich der Variablen angegeben. Die vorgegebenen Werte der Variablen sind 0 und 1 und werden mittels

der Formeln 4.3 definiert.

$$\begin{aligned} X_{Artikel\ ,\ Gutachter} &\geq 0 \\ X_{Artikel\ ,\ Gutachter} &\leq 1 \end{aligned} \tag{4.3}$$

Die in der Gleichung 4.4 definierte Restriktion dient zur Festlegung der genauen Anzahl der Gutachter für einen Artikel und dadurch zu einer vorgegebenen Anzahl von Gutachten pro Artikel.

$$\sum_{j}^{Gutachter} X_{Artikel_j} = Gutachten\ pro\ Artikel \tag{4.4}$$

Der letzte Restriktionstyp berücksichtigt die Arbeitsauslastung jedes einzelnen Gutachters. Da eine Reihe von Konstellationen auftreten können, zu denen keine Lösung berechnet werden kann, wird eine Benutzereingabe einer Abweichung von der gewünschten Anzahl von Artikeln pro Gutachter zugelassen. Aufgrund dieser Abweichung sind die Ungleichungen 4.5 definiert zu:

$$\begin{aligned} \sum_{i}^{Artikel} X_{Gutachter_i} &\geq untere\ Arbeitslast \\ \sum_{i}^{Artikel} X_{Gutachter_i} &\leq obere\ Arbeitslast \end{aligned} \tag{4.5}$$

Soll auf Basis der in Abbildung 4.11 dargestellten Begutachtungswünsche der Gutachter eine gerechte und adäquate Verteilung automatisch berechnet werden, ist im ersten Schritt mittels der erläuterten Gleichungen die Eingabedatei für den lp_solve zu generieren (siehe Abbildung 4.12).

Hierzu werden alle Einträge aus der Biddingmatrix berücksichtig. Es wird z.B. das Bidding von Gutachter *G1* für Artikel *001* mit der Gleichung $X\_001\_1 = 1$ festgelegt und der Konflikt zwischen *G3* und *001* geht als Bedingung mit $X\_001\_3 = 0$ in die Optimierung ein. Durch die Übernahmen der in der Biddingmatrix spezifizierten Randbedingungen ist es möglich, eine zusätzliche Verteilung durchzuführen. Das inkrementelle Anpassen einer berechneten Verteilung ist in der Praxis von größter Bedeutung, da sonst der komplette Verteilungsprozess zurückgenommen werden müsste. Dieses wäre für die Wissenschaftsgemeinde, insbesondere die Gutachter, nicht akzeptabel. $\geq$ und $\leq$

```
min: 11 X_001_1 + 20 X_001_2 + 10 X_001_4 + 20 X_002_1 + 20 X_002_2 + 30 X_002_3 +
     20 X_002_4 + 30 X_003_1 + 30 X_003_2 + 21 X_003_3 + 10 X_003_4 + 11 X_004_2 +
     21 X_004_3 + 10 X_004_4;
X_001_1 = 1;
X_001_2 > 0; X_001_2 < 1;
X_001_3 = 0;
X_001_4 > 0; X_001_4 < 1;
X_002_1 > 0; X_002_1 < 1;
X_002_2 > 0; X_002_2 < 1;
X_002_3 > 0; X_002_3 < 1;
X_002_4 > 0; X_002_4 < 1;
X_003_1 > 0; X_003_1 < 1;
X_003_2 > 0; X_003_2 < 1;
X_003_3 = 1;
X_003_4 > 0; X_003_4 < 1;
X_004_1 = 0;
X_004_2 = 1;
X_004_3 = 1;
X_004_4 > 0; X_004_4 < 1;
X_001_1 + X_001_2 + X_001_3 + X_001_4 = 3;
X_002_1 + X_002_2 + X_002_3 + X_002_4 = 3;
X_003_1 + X_003_2 + X_003_3 + X_003_4 = 3;
X_004_1 + X_004_2 + X_004_3 + X_004_4 = 3;
X_001_1 + X_002_1 + X_003_1 + X_004_1 > 3;
X_001_1 + X_002_1 + X_003_1 + X_004_1 < 3;
X_001_2 + X_002_2 + X_003_2 + X_004_2 > 3;
X_001_2 + X_002_2 + X_003_2 + X_004_2 < 3;
X_001_3 + X_002_3 + X_003_3 + X_004_3 > 3;
X_001_3 + X_002_3 + X_003_3 + X_004_3 < 3;
X_001_4 + X_002_4 + X_003_4 + X_004_4 > 3;
X_001_4 + X_002_4 + X_003_4 + X_004_4 < 3;
int X_001_1, X_001_2, X_001_3, X_001_4, X_002_1, X_002_2,
X_002_3, X_002_4, X_003_1, X_003_2, X_003_3, X_003_4, X_004_1, X_004_2, X_004_3, X_004_4;
```

Abbildung 4.12: Solver-Input - Zielfunktion, Restriktionen, Variablendeklaration

entsprechen in der lp_solve Syntax > und <. In der ersten Zeile der Datei wird die Zielfunktion definiert, gefolgt von den Restriktionen mit denen das Minimum (min:) zu berechnen ist. Die letzte Zeile definiert den Typ der einzelnen Variablen, hier Integer.

Im nächsten Schritt wird dem Solver die Eingabedatei übergeben. Der Solver wird als Thread gestartet und blockiert den eigentlichen Workflow nicht. Das Ergebnis der Optimierung wird in ein vorgegebenes Format als eine Datei gespeichert, siehe Abbildung 4.13.

Die Datei enthält die Variablen der Zielfunktion mit den entsprechenden berechneten Werten, die durch Restriktionen auf 0 oder 1 festgelegt wurden. Eine 1 bedeutet eine Delegierung. Das Ergebnis wird in Abbildung 4.14 dargestellt.

Es ist ersichtlich, dass eine ausgeglichene Verteilung erzielt wurde. Jeder Artikel ist laut Vorgabe an drei Gutachter verteilt worden und die Arbeitsaus-

```
X_001_1          1
X_001_2          1
X_001_3          0
X_001_4          1
X_002_1          1
X_002_2          1
X_002_3          1
X_002_4          0
X_003_1          1
X_003_2          0
X_003_3          1
X_003_4          1
X_004_1          0
X_004_2          1
X_004_3          1
X_004_4          1
```

Abbildung 4.13: Solver-Output - Lösung der Zielfunktion

ID	G1	G2	G3	G4	ID	Sum:(3)
001	B ☑	I ☑	C	I ☑	001	3
002	I ☑	I ☑	I ☑	U ☐	002	3
003	U ☑	U ☐	B ☑	I ☑	003	3
004	C	B ☑	B ☑	I ☑	004	3
Sum:	**3**	**3**	**3**	**3**		

Abbildung 4.14: Bidding-Matrix - Lösungsvorschlag des lp-solve

lastung unter den Gutachtern ist identisch. Konflikte wurden während der Berechnung beachtet und die Biddings in ihrer Gewichtung eingeplant. So wurden **B** vor **I** und **I** vor **U** berücksichtigt. Die Konstellation von der Anzahl der Artikel und Gutachter zur Anzahl der Zuweisungen von drei Gutachtern pro Artikel war in diesem Beispiel trotz zweier Konflikte lösbar.

Abbildung 4.15 hingegen zeigt eine durch das Einfügen des Konfliktes zwischen Artikel *001* und Gutachter *G4* veränderte Konstellation. Unter denselben Annahmen, dass jeder Artikel von drei Gutachtern beurteilt wird, findet lp_solve keine Lösung für dieses Problem. In diesem Fall kommt die Abweichung zum Tragen, indem einen prozentuale Abweichung über die GUI der Konferenz angegeben wird, die sich in den Restriktionen 4.5 auswirken. Die untere und obere Auslastung wird entsprechend der Abweichung nach unten bzw. oben korrigiert. Für das betrachtete Beispiel wird eine Lösung bei einer 35%-igen Abweichung erreicht.

Die berechneten Verteilungsvorschläge können manuell angepasst und zwi-

ID	G1	G2	G3	G4	ID	Sum:(3)
001	B ☑	I ☑	C	C	001	2
002	I ☑	I ☑	I ☑	U ☐	002	3
003	U ☑	U ☐	B ☑	I ☑	003	3
004	C	B ☑	B ☑	I ☑	004	3
Sum:	3	3	3	2		

Abbildung 4.15: Bidding-Matrix - Lösungsvorschlag mit 35%-iger Abweichung

schengespeichert werden. Das Feature ermöglicht eine inkrementelle Anpassung der Planung und den wiederholenden Einsatz der Verteilungsberechnung auf Basis der veränderten Randbedingungen. Sobald die Verteilung feststeht, kann die Zuweisung automatisiert durchgeführt werden. Jeder Gutachter wird per Email über die neuen Begutachtungsaufgaben informiert und die einzelnen Zuweisungen als eine neue Aufgabe in der Arbeitsliste (tasklist) des entsprechenden Gutachters hinterlegt. Bereits existierende Delegierungen werden ebenfalls in der Bestimmung einer optimalen Verteilung berücksichtigt, sodass auch in der Begutachtungsphase mittels des Bidding Features zusätzliche Verteilungen erfolgen können.

Der in [Tay08] vorgestellte Ansatz führt die Verteilung von Begutachtungsaufgaben ebenfalls auf ein lineares Optimierungsproblem zurück, allerdings berücksichtigt die beschriebene Vorgehensweise keine Begutachtungsvorschläge (biddings) von den eigentlichen Gutachtern. Die Konferenzleiter schlagen unter Angabe von Gewichtungen Zuweisungen zwischen Artikeln und Gutachtern vor. Dieses verursacht eine zusätzliche Belastung der Konferenzleitung und entspricht nicht dem eigenen Ansatz, der die Konferenzleitung während der Verteilung der Begutachtungsaufgaben entlastet und die Gutachter in die Bestimmung einer optimalen Verteilung integriert. Werden komplexere Konferenzen mit einer großen Anzahl von Artikeln und Gutachtern betrachtet führt der Ansatz aus [Tay08] bei einer kleinen Konferenzleitung zu einem immensen Aufwand. Des Weiteren stellt die berechnete Verteilung mathematisch zwar eine optimale Lösung dar, allerdings stoßen Zwangszuweisungen von Begutachtungsaufgaben aus eigener Erfahrung oftmals auf Ablehnung, was im Umkehrschlußss zu einer erneuten Verteilung führen kann.

4.4 Konfigurierbarer Bewertungsworkflow

Gutachten dienen innerhalb von Konferenzen zur Beurteilung und Diskussion von Artikeln. Die Gutachten werden auf der Basis definierter Beurteilungsformulare durchgeführt, die einem für die Konferenz vorgeschriebenen Muster entsprechen. Die wiederkehrende Anforderung der Anpassungen der Begutachtungsformulare legte den Grundstein für das Design und die Umsetzung eines in seiner Konfiguration flexiblen Begutachtungssystems, dessen Formulare in Hinsicht auf die Elemente und Formeln zur Berechnung von Benotungen frei konfigurierbar sind. Elemente wie zum Beispiel *check boxes* oder *radio buttons* können mit Werten hinterlegt und in die Formel zur Berechnung einer Note einbezogen werden.

Die Abbildung 4.16 zeigt die Konfigurationsseite zur Erstellung bzw. Modifikation von Begutachtungsformularen (report forms).

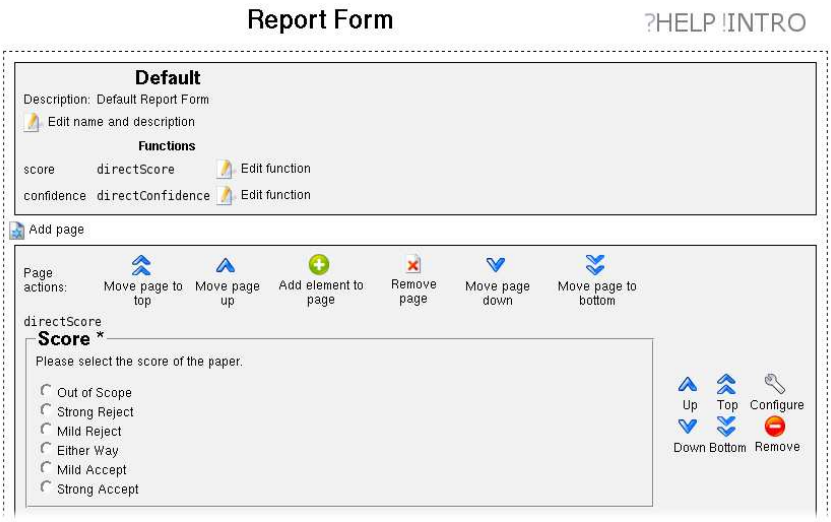


Abbildung 4.16: Reportformular - Konfiguration des Formulars

Das Formular besteht aus den Basiselementen Formularname, einer Beschreibung des Formulars, der Benotung (score) und der Selbsteinschätzung des Begutachters (confidence). Die Selbsteinschätzung bezieht sich auf die fachliche Kompetenz des Gutachters im Bezug auf das Thema des zu begutachtenden Artikels. Während der Bestimmung der Gesamtnote eines Artikels, die in der Regel von drei Gutachten abhängig ist, gehen die einzelnen Benotungen der Gutachter gewichtet nach ihrer Fachkompetenz ein.

Zur Erstellung eines Begutachtungsformulars stehen verschiedene Formularelemente zur Verfügung:

- **Text:** Beliebige Informationen können in textlicher Form im Formular platziert werden.
- **Text Area:** Der Dienstbenutzer kann einen längeren Text eingeben, wobei die Länge des Textes definiert werden kann.
- **Radio Buttons:** Aus einer Menge von z.B. vorformulierte Antworten kann eine ausgewählt werden.
- **Check Boxes:** Mehrere Selektierungen sind möglich, wodurch z.B. ein Gutachter eine Frage mit mehreren Auswahlmöglichkeiten beantworten kann.
- **Select Element:** Ermöglicht eine Auswahl eines Elements aus einer Liste.
- **File Element:** Dieses Element hängt eine Datei an das Begutachtungsformular an. Der Typ des Files und dessen Größe sind definierbar.

Durch Kombination der obigen Elemente wird das Formular erstellt. Das Formular kann sich über mehrere Seiten erstrecken und die einzelnen Elemente beliebig oft in einem Formular verwendet werden. Die Elemente selbst sind für ihre jeweilige Anwendung konfigurierbar. Sie können als *required* markiert werden, um die Verwendung des Elementes durch den Gutachter verpflichtend zu machen. Dieses könnte eine Antwort auf eine Qualitätsfrage des Artikels oder die Eingabe der schriftlichen Beurteilung des Artikels in einem Textfeld sein. Eine Einreichung des ausgefüllten Formulars wird nur zugelassen, sobald alle verpflichtenden Elemente bearbeitet wurden. Die Annotation *confidential* kennzeichnet ein Element als vertraulich und ist in Bezug auf das Rollen/Rechte-Management nur für Benutzer verfügbar, deren Benutzerrollen dieses Feature beinhalten.

In Abbildung 4.17 wird das Element *Radio Buttons* gezeigt, das in der momentanen Verwendung für die Erstellung des *Score*-Abschnittes in dem Begutachtungsformular zum Einsatz kommt. Das *Label* ist der Name, unter dem das definierte Element im Begutachtungsformular erscheint. Die *Instruction* gibt dem Benutzer, der das Formular ausfüllt, Hinweise und Anweisungen. Die Anzahl, Benennung und Wertzuweisung kann frei gewählt werden. Eine Vorselektion eines Eintrages sowie die Ausrichtung des Textes kann konfiguriert werden.

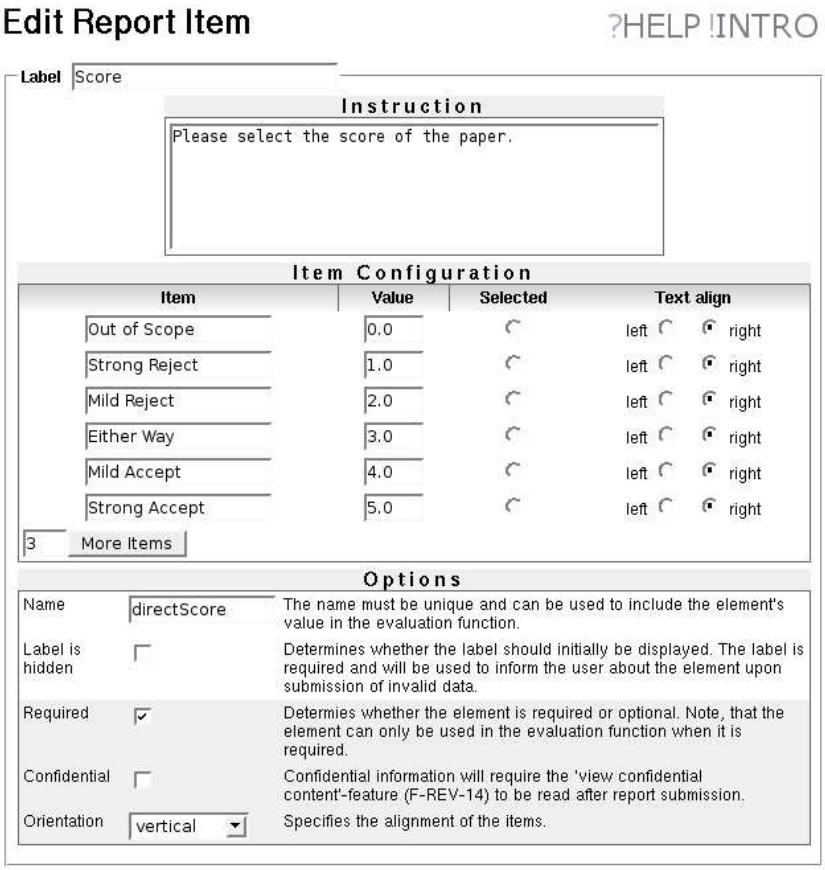


Abbildung 4.17: Reportformular - *Radio Buttons* Element

Ein wichtiger Aspekt ist die eindeutige interne Benennung des Elementes (Name). Die Werte der Elemente sind in der konfigurierbaren Berechnungsformel für den *Score* bzw. *Confidence* verwendbar. Durch diese Vorgehenweise ist es möglich, die Benotung des Artikels von mehreren Elementen eines Formulars abhängig zu machen.

Eine HTML-Repräsentation eines Begutachtungsformulars zeigt Abbildung 4.18 auf. Dieses Standardformular besteht aus jeweils zwei *Radio Buttons* und *Text Area* Elementen. Die Markierung * hebt die Pflichtelemente hervor, die vom Gutachter bearbeitet werden müssen. Das Element *Comments for PC Eyes only* ist ein vertrauliches Feld, dessen Inhalt nur ausgewählten Rollen der Konferenz angezeigt wird. In diesem Fall dient es für zusätzliche Bewertungen, die für andere Gutachter, aber auf keinen Fall für die Autoren bestimmt sind.

Im Falle des OCS existieren zwei Typen von Begutachtungsformularen, die je-

weils neu oder auf Basis von vordefinierten Formularen erstellt werden können:

- **Single Report:** Die vom PC Member verfasste Beurteilung.
- **Final Report:** Die abschliessende Beurteilung auf Basis der *Single Reports* und des PC-Meetings bzw. der Forendiskussion.

Abbildung 4.18: Reportformular - Gerendertes Formular

Von beiden lassen sich beliebig viele Reportformulare realisieren. Die Beziehung der Formulare zu den Artikeln wird durch die Assoziation zwischen dem

Reportformular und einer oder mehrerer Artikelkategorien hergestellt. Aufgrund der Einteilungen der Artikel kann zu jeder Gruppe von Artikeln ein eigenes Begutachtungsformular definiert und verwendet werden, siehe Abbildung 4.19.

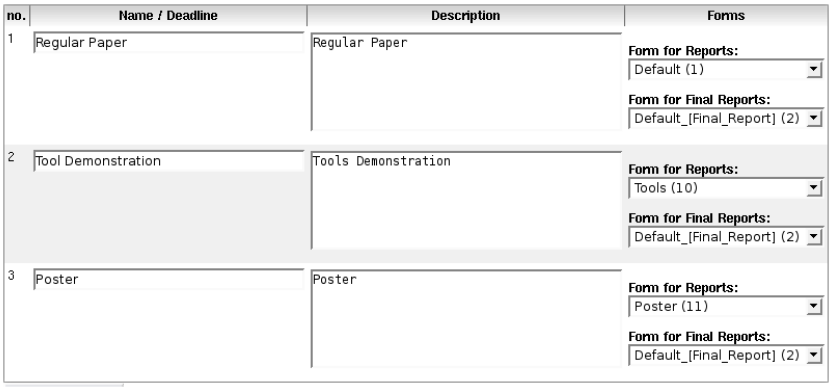


Abbildung 4.19: Reportform - Kategorienzuweisung

Jeder Kategorie wurde ein eigenes *Single Report*-Formular zugeteilt, wobei ein gemeinsames Formular für den *Final Report* verwendet wird.

Die Realisierung der Begutachtungsformulare basiert auf XML [5]. XML ist eine Auszeichnungssprache zur strukturierten Beschreibung von Daten in einer lesbaren Form, z.B. als ASCII [6]. Das W3C [7] [Con09] gab 1998 eine XML-Spezifikation heraus, die XML standardisierte. XML-Dateien lassen sich mittels einer DTD [8] oder einer XSD [9] auf ihre Wohlgeformtheit validieren. Die Definitionen spezifizieren die Struktur und die erlaubten Elemente für eine XML-Datei. Für dieses Feature wurde das XML Schema verwendet, da es unter anderem im Gegensatz zur DTD mehr und komplexere Datentypen bietet.

Für die Bearbeitung der XML-Dateien wird das Framework JAXB-2.x [10] eingesetzt [mic09h] [mic09i]. Dieses ermöglicht einerseits XML-Dokumente als Java-Objekte abzubilden (unmarshal) und andererseits Java-Objekte wiederum in XML zu überführen (marshal). Abbildung 4.20 (aus [mic09i]) zeigt die Arbeitsweise von JAXB.

[5] Extensible Markup Language
[6] American Standard Code for Information Interchange
[7] World Wide Web Consortium
[8] Document Type Definition
[9] XML Schema Definition
[10] Java Architecture for XML Binding

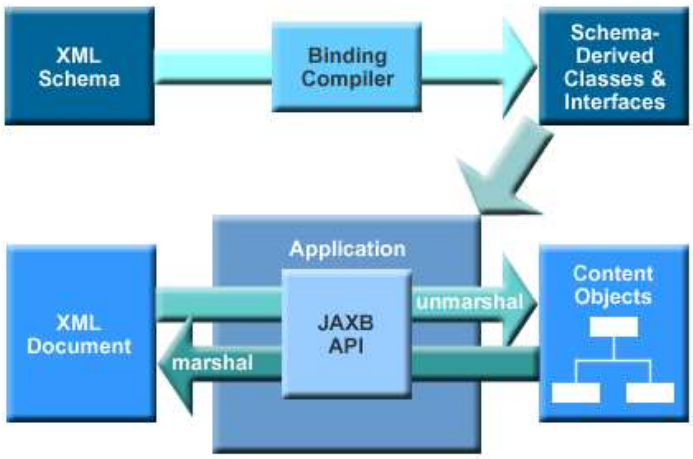


Abbildung 4.20: JAXB - Funktionsweise

Als vorbereitender Schritt werden mit Hilfe des *Binding Compilers* (xjc) aus dem XML Schema die Java-Klassen (.java) generiert. Diese Klassen stellen die Java-Repräsentation des XML Schemas dar. Die Klassen müssen anschließend kompiliert werden (javac), um für das JAXB-Framework zur Laufzeit der Applikation zur Verfügung zu stehen. XML-Dokumente die in Bezug auf das Schema wohlgeformt sind, können mittels dem JAXB Binding Framework durch Instanzen der zuvor generierten Java-Repräsentationen des XML Schemas abgebildet werden. Modifikationen können nun an der Java-Repräsentation des XML-Dokuments durchgeführt werden. Basierend auf den Java-Repräsentationen lassen sich mit JAXB die entsprechenden XML-Dokumente erzeugen.

Im OCS werden die Gutachten als XML-Dokumente in der Datenbank persistent gehalten. Ist ein Gutachten zu manipulieren, wird dieses aus der Datenbank gelesen und mittels JAXB zu Java-Objekten transformiert. Um das Gutachten zu editieren, wird mit einem XSLT [11]-Stylesheet das XML-Dokument nach HTML übersetzt. Die auf der GUI-Ebene durchgeführten Änderungen werden an die Java-Repräsentation weitergeleitet. Die Java-Objekte werden zur Speicherung mittels JAXB nach XML transformiert und in der Datenbank gespeichert.

[11] Extensible Stylesheet Language Transformation

Teil III

Das Entscheidungssystem OCS

Kapitel 5

Workflows und Features

Dieses Kapitel dient zur detaillierteren Vorstellung der Workflows und Features des betrachteten Referenzsystems OCS. Für bzw. durch den OCS sind unter anderem die eingeführten Konzepte (siehe Kapitel 4) entstanden und finden in der Praxis ihren Einsatz. Der *Online Conference Service* ist ein Web-basiertes Entscheidungssystem für die Begutachtung von wissenschaftlichen Artikeln, das die an einer Konferenz beteiligten Personen in jeder Phase einer Konferenz in ihrer gemeinschaftlichen Arbeit unterstützt und sich wiederholende Geschäftsprozesse automatisiert. Die für eine Begutachtung im Rahmen einer Konferenz verantwortlichen Personen werden unter dem Begriff Programmkomitee zusammengefasst. Die Komiteemitglieder stellen Benutzer des OCS's dar, die im OCS unterschiedliche Rollen bekleiden können. Diese sind für das Komitee insbesondere die Rollen *PC Chair* und *PC Member*. Wobei die Rolle *PC Chair* die Konferenzleiter und die *PC Member* Rolle die Gutachter der Konferenz repräsentieren. Ferner existieren weitere vordefinierte Rollen wie Author, CoAuthor, SubReviewer und Administrator (siehe Tabelle 1.1). Deren Aufgabenbereiche werden im Laufe des Kapitels am Geschäftsprozess des OCS näher erläutert.

Einem Dienstbenutzer können mehrere Rollen zugeordnet sein. Diese verleihen ihm in Abhängigkeit der jeweiligen aktiven Rolle eine personalisierte Sicht auf den OCS, dessen Funktionalitäten und der zu bearbeitenden Objekte, wie z.B. einen Artikel. Die Konfiguration und Zuordnung der Rollen sowie die direkte Rechtevergabe auf Ressourcen kann mittels des Rollen/Rechtesystems feingranular eingestellt und während der laufenden Konferenz manipuliert werden (siehe Kapitel 4.2).

5.1 Konferenzphasen

Das in Abbildung 5.1 dargestellte Kontrollflussdiagramm gibt einen vereinfachten Überblick der im OCS existierenden Konferenzphasen.

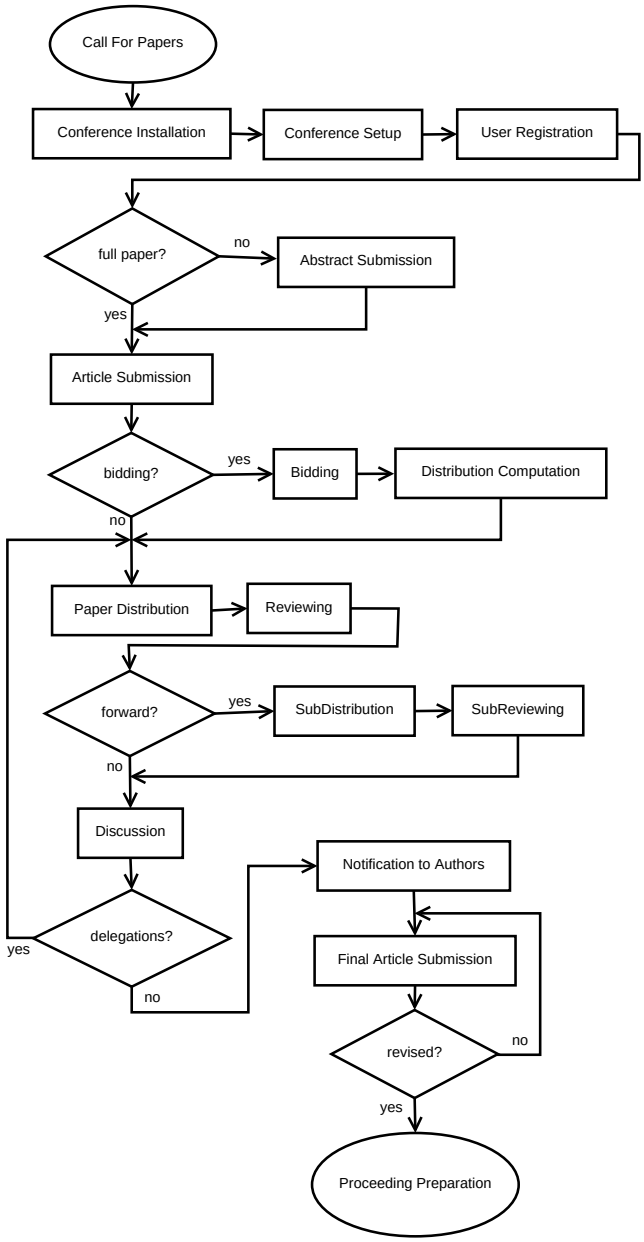


Abbildung 5.1: OCS - Kontrollflussdiagramm der Konferenzphasen

Eine Konferenz beginnt neben der Organisation von Tagungsräumen und der Zusammenstellung der zu buchenden Hotels mit dem **Call For Paper** (CFP). Dieses ist der Aufruf an Personen, ihre Forschungsergebnisse in Form eines Artikels zur Konferenz einzureichen. Angesprochen werden in der Regel Forscher aus dem akademischen und industriellen Bereich. Nach dem frühzeitigen Versenden bzw. Aushang des CFP durch die Konferenzorganisatoren, in dem alle relevanten Konferenzinformationen insbesondere die Internetadresse des verwendeten Begutachtungssystems bekannt gegeben wird, beginnt die Installation und Einrichtung des Begutachtungssystems. Die Installation des Systems erfolgt in der Regel drei bis vier Wochen vor Beginn der Einreichungsphase und wird von dem jeweiligen Anbieter vorgenommen. Dies ist im Falle des OCS, der für die LNCS-Reihe zur Verfügung gestellt wird, der Springer Verlag Heidelberg.

Die Konfiguration der Konferenz wird in der Phase **Conference Setup** vom PC Chair mit Hilfe des *Setup-Features* durchgeführt. Der Dienst wird in Bezug auf sein Verhalten und Aussehen auf die zu unterstützende Konferenz angepasst. Zeitfristen (deadlines), Rollen, Reportformulare und artikelspezifische Seiten können konfiguriert werden. Des Weiteren wird eine minimale Registrierung des Komitees durchgeführt, dessen Mitgleider die ihre Registrierung zum Dienst vervollständigen und bestätigen müssen. Die getätigten Konfigurationen sind über die **Conference Setup** Phase hinaus manipulierbar.

Während der **User Registration** Phase wird das Registrieren der *PC Members* durch den *PC Chair* und die Selbstregistrierung der *Authors* zusammengefasst. Eine selbständige Registrierung ist im OCS mit der Zuweisung der Rolle *Author* verbunden. Diese Rolle beinhaltet eine eingeschränkte Sicht auf die Dienstfunktionalitäten. Alle anderen Rollen müssen vom PC Chair explizit an Benutzer vergeben werden.

In der Phase **Abstract Submission** und **Article Submission** ist es den registrierten und authentisierten Autoren erlaubt, ihre Beiträge einzureichen. Beide Phasen werden durch das *Article-Feature* abgedeckt. Die Einreichung kann entweder direkt in Form des vollständigen Artikels oder durch eine kurze Artikelzusammenfassung (abstract) geschehen. Die *Abstract Submission* zieht die *Article Submission* nach sich und dient den Konferenzleitern zur Einschätzung des Konferenzumfanges und zur ersten Planung der Begutachtungsverteilung. Die *Abstract Submission* Phase ist über das *Setup* der Konferenz ein bzw. ausschaltbar, indem das entsprechende Dienst-Feature für die Rolle *Author* aktiviert bzw. deaktiviert wird. Durch die Aktivierung oder Deaktivierung von Features unter Verwendung des bereits erörterten Rollen/Rechtesystems lässt sich der Kontrollfluss zur Laufzeit verändern, wobei die Features zu Beginn

der Konferenz existieren und während der Konferenz statisch sind.

Die **Bidding** und **Distribution Computation** Phasen dienen zur Planung und Festlegung der Artikelverteilungen an die Gutachter und sind optionale Features, die mittels des *Setup-Features* dem Workflow hinzugeschaltet werden können. Die Gutachter stimmen für die Artikel, die sie begutachten wollen und geben Desinteresse oder Konflikte an. Die **Paper Distribution** Phase weist den Gutachtern die Begutachtungsaufgaben zu. Die drei Phasen werden vom *Bidding-Feature* realisiert (siehe Kapitel 4.3).

Sobald die Artikel zugewiesen sind, beginnt die **Reviewing** Phase, die die Begutachtung der Artikel darstellt. Gutachter verfassen ihre Gutachten entweder alleine oder sie bauen ihr Gutachten auf Gutachten von *SubReviewern* auf. Die Weiterleitung ist optional und beinhaltet die Phasen **SubDistribution** und **SubReviewing**. Die Entscheidung über die Annahme oder Ablehnung eines Artikels findet in der **Discussion** Phase statt, die aus dem *Forum* und *Evaluation Feature* besteht. Diese kann entweder mittels der Artikelforen oder physikalisch stattfinden. Beim PC-Treffen wird die Diskussion durch die *EvaluationMatrix* unterstützt. Die einzelnen Ergebnisse werden durch Statusänderungen der Artikel im Dienst eingepflegt und die jeweilige Entscheidung in der Matrix graphisch hervorgehoben.

Für Artikel, die sich im Zustand *angenommen* oder *abgelehnt* befinden, startet die **Notification to Author** Phase. Konferenzleiter schreiben ein Gutachen, das das Gesamtergebnis der Begutachtung beinhaltet. In der Regel findet dieser Prozess für die akzeptierten und abgelehnten Artikel jeweils gleichzeitig statt. Autoren, deren Artikel angenommen wurden, reichen ihre überarbeitete Artikelversion ein, die bei Erfüllung der Gutachterauflagen zur Veröffentlichung zugelassen wird.

In der **Proceeding Preparation** Phase wird der Tagungsband erstellt und an dem Verlag zum Druck weitergeleitet. Hier kommt das *Proceeding-Feature* zum Tragen, das den PPS [Hol06] widerspiegelt (siehe Kapitel 7.3).

5.2 Lebenszyklus eines Artikelobjektes

Im Folgenden wird detaillierter auf das Artikelobjekt, das die zentrale Rolle im Begutachtungsprozess einnimmt, eingegangen. Die Zugriffe auf die gemeinschaftlich zu bearbeitenden Objekte, wie z.B. die Artikel, sind zustandsabhängig. D.h. der Zustand eines Artikels ändert sich zur Laufzeit der Konferenz dynamisch, in Abhängigkeit von auftretenden Ereignissen (events), die während der im Kapitel 5.1 beschriebenen Konferenzphasen stattfinden. Der

Lebenszyklus des Artikelobjektes in Form von Zuständen und deren Übergängen (translations) ist in Abbildung 5.2 dargestellt.

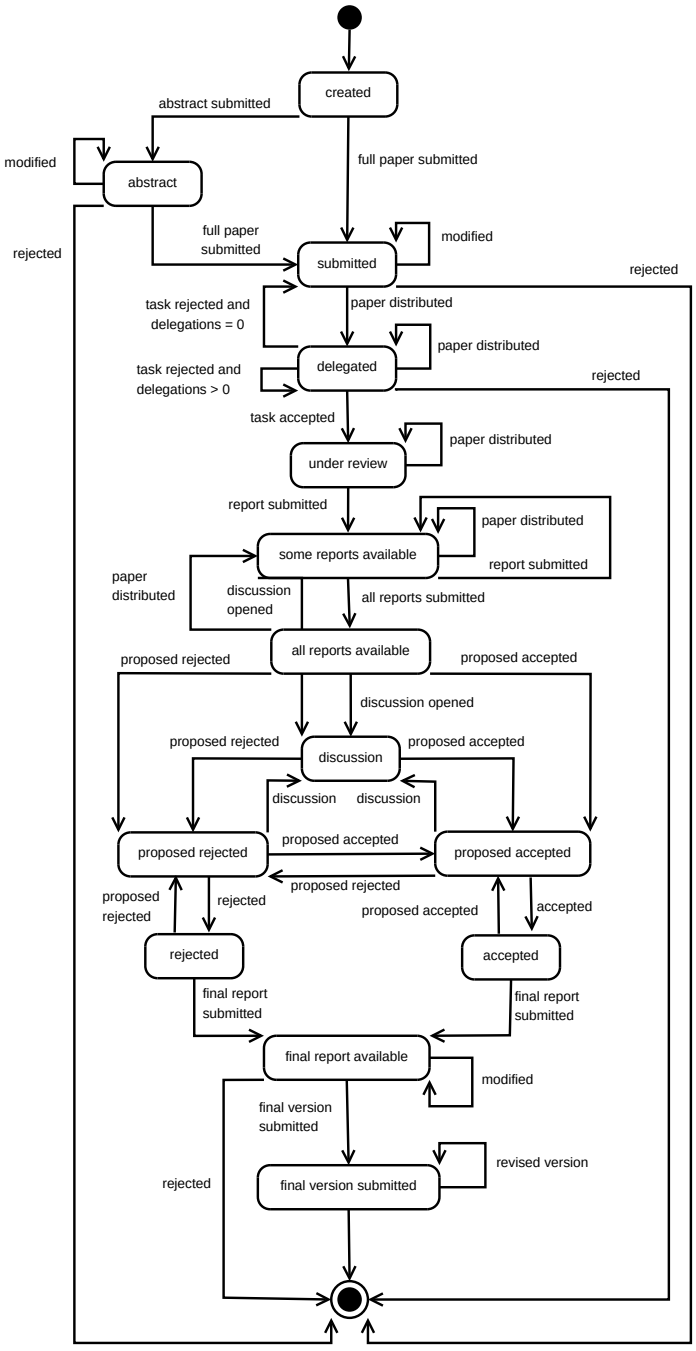


Abbildung 5.2: Artikelzustände im Begutachtungsprozess

Die einzelnen Zustände sind in dem Zustandsdiagramm mit einem Rechteck dargestellt, wobei die Zustandsübergänge von einem Zustand in den nächsten durch einen Pfeil repräsentiert werden, der durch das auslösende Ereignis des Überganges annotiert ist. Die Ereignisse sind Aktionen von Dienstbenutzern, die mittels Dienstfunktionalitäten durchgeführt werden. Die dynamisch wechselnden Zustände bestimmen die erlaubten Aktionen auf einen Artikel. Jedem Zustand werden mittels einer Propertydatei Features zugeordnet (siehe Kapitel 4.2.2). Der Zugriffskontrollmechanismus des OCS vergleicht dieses Feature mit den benötigten Features für eine Aktion, d.h. für die Verwendung einer Dienstkomponente, um z.B. den Artikel zu manipulieren, mit den erlaubten Features des Dienstbenutzers.

Sobald ein Autor mit der Einreichung seines Artikels beginnt, wird ein Artikelobjekt instanziiert, das den Zustand **created** aufweist. Wird die Einreichung vom Autor abgeschlossen, ist der Zustand des Artikels entweder **abstract** oder **submitted**. Dieses ist abhängig davon, ob es sich bei der Einreichung nur um eine Ankündigung oder einen vollständigen Artikel handelt.

Die Verteilung eines Artikels zur Begutachtung führt in den Zustand **delegated**, dieser beinhaltet die Features für das Schreiben von Gutachten. Aktionen durch den PC Member bestimmen die Übergänge von **delegated** zu **under review**, anschliessend zu **some reports available** und **all reports available**. Die Zustände zeigen die Akzeptanz des Gutachtenauftrages, die teilweise und schlussendlich die komplette Einreichung aller Reports auf und geben unterschiedliche Aktionen frei. Ab **some reports available** ist es bereits für den PC Chair möglich, den Artikelzustand auf **discussion** zu setzen, sodass die Gutachter mit den bislang eingereichten Gutachten die Diskussion beginnen können. Gutachter mit einem ausstehenden Gutachten hingegen erlangen erst Zugriff auf das Artikelforum, wenn sie ebenfalls ihr Gutachten eingereicht haben. Dadurch wird eine eventuelle Beeinflussung durch das Lesen der sich bereits im Dienst befindenden Gutachten vermieden.

Vom **all reports available** ausgehend ist der PC Chair in der Lage, den Zustand auf **discussion**, **proposed accepted** oder **proposed rejeced** zu setzen. Die Zustände **accepted** und **rejected** bilden das jeweilige Endergebnis der Diskussion. Während der Diskussionphase kann der PC Chair zwischen den Zuständen **discussion**, **proposed accepted**, **proposed rejected**, **accepted** oder **rejected** wechseln. Dadurch wird direkt die momentane Diskussionssituation in der *EvaluationMatrix* für das Programmkomitee visualisiert.

Nach Einreichung des *final reports* durch den PC Chair nimmt der Artikel den Zustand **final report available** an. Abschließend wird der Zustand auf **final version submitted** gesetzt, nachdem der Autor seine überarbeitete Artikel-

version hochgeladen hat. Das erneute Einreichen einer verbesserten Version verändert nicht den Artikelstatus.

Die Beschreibung hat die Zustandsübergänge fokussiert auf einen Begutachtungsdurchgang eines Artikels aufgezeigt. Im Lebenszyklus eines Artikels treten auch Ereignisse auf, die zu keiner Veränderung des Zustands oder zu einer Rücksetzung in einen früheren Zustand führen (siehe Abbildung 5.2).

5.3 OCS Features

Die Geschäftsprozesse des OCS sind graphbasiert (siehe Kapitel 3.1.1) modelliert. Die Funktionalitäten der Geschäftslogik, die für ein bestimmtes Objekt realisiert wurden, sind zu einem Hauptfeature (siehe Kapitel 4.1) gruppiert. Features repräsentieren die Kategorien der Dienstfunktionalitäten, die für einen Benutzer zur Laufzeit über eine Navigationsleiste verfügbar sind (siehe Abbildung 4.1). Im Folgenden werden die wichtigsten Hauptfeatures und deren Funktionalitäten kurz vorgestellt.

Article

Der Begutachtungsprozess eines Konferenzbeitrages beginnt mit der Einreichung eines Artikel und endet mit dem Hochladen der finalen Artikelversion. Das *Artikel-Feature* stellt eine Reihe von Geschäftsprozessen und deren Komponenten zur Bearbeitung und Verwaltung der Artikel zur Verfügung. Die Einreichung umfasst die Teilprozesse (SubFeatures) *Abstract Submission*, *Modify Abstract*, *Article Submission*, *Modify Article*, *Submit Article*, *Submit Final Version* und *Modify Final Version*. Artikel können heruntergeladen, gelesen, modifiziert, gespeichert, gesichert, wieder eingespielt und gelöscht werden. Die Teilprozesse lassen sich feingranular Rollen zuordnen. So ist z.B. das Lesen von Artikeln mit einem zusätzlichen Filter versehen, der ermöglicht, zwischen eigenen, delegierten und involvierten Artikeln zu differenzieren. Des Weiteren verwaltet es die Artikelzustände und ist zuständig für die Zustandsänderungen und ihre Plausibilitätsprüfung.

Bidding

Das Bidding-Feature unterstützt die Konferenzleiter bei der Verteilung von Begutachtungsaufgaben. Es können einerseits Gutachterwünsche in Form einer Artikelwahl berücksichtigt, andererseits unter Verwendung des externen

Programms `lp_solve` eine Verteilung berechnet werden. Die detaillierte Beschreibung ist in Kapitel 4.3 zu finden.

Conflict

Dieses Feature verwaltet die Konflikte zwischen den Artikeln und den Dienstbenutzern. Konflikte treten auf, wenn z.B. ein Gutachter einen Autor sehr gut kennt, dann darf dieser kein Gutachten bzgl. des Artikels schreiben oder die Begutachtung verfolgen. Die Gefahr der Beeinflussung wäre gegeben. Konflikte können von den Konferenzleitern jederzeit und von den Gutachtern in der Bidding-Phase gesetzt werden. Die in der Konfliktmatrix festgelegten Konflikte finden Berücksichtigung bei der Sicht auf den Dienst und der Erlaubnis, Dienstfunktionalitäten zu verwenden. In erster Linie wird der Zugriff auf die relevanten Gutachten, die Einträge in der EvaluationMatrix und die betroffenen Foren verweigert.

Deadline

Um einen zeitlich gesicherten Ablauf der Konferenz zu gewährleisten, können Zeitfristen für Aufgaben hinterlegt und überwacht werden. Es lassen sich unter anderem Zeitfristen für die Artikel- und Report-Einreichungen spezifizieren. Das Feature wirkt unterstützend für den Konferenzleiter, der durch eine Mitteilung per Email über den Ablauf einer Frist informiert wird. Dieser kontaktiert den betroffenen Dienstbenutzer persönlich. Die Erfahrung hat gezeigt, das automatisch gesendete Emails an Dienstbenutzer wenig Erfolg haben, da sie einfach zu unpersönlich wirken.

Delegation

Das *Delegation Management* setzt sich zusammen aus den Geschäftsprozessen zur Zuteilung der Begutachtungsaufgaben und der *DelegationMatrix* einer Monitoring-Komponente für Delegierungen. Die jeweiligen Deadlines einer Delegierung können modifiziert, und abgelehnte Begutachtungsaufgaben neu verteilt werden. Jede Delegierung unterliegt zwei Deadlines, der *Decision* und *Task*-Deadline. Die erste Deadline ist der Zeitpunkt, bis wann der Gutachter die Aufgabe bestätigt haben soll und die Zweite zeigt die Zeit auf, bis zu der das Gutachten eingereicht sein muss. Die Deadlines werden vom Deadline Management überprüft und im Falle einer Zeitüberschreitung die Konferenzleiter benachrichtigt. Die Weiterleitung des Begutachtungsauftrages (SubReviewing)

an Subreviewer wird ebenfalls von diesem Feature abgedeckt.

Email

Dieses Feature ermöglicht das Verfassen und Senden von Emails. Hierzu können Textbausteine sowie Emailvorlagen definiert werden. Die Benutzung von Tags erleichtert zusätzlich das Schreiben von Emails. Mails können gleichzeitig an eine Gruppe von Dienstbenutzern gesendet werden, z.B. an alle PC Member, Autoren oder nur an alle Autoren der akzeptierten Artikel. Durch die erwähnten Tags ist es möglich, eine Emailvorlage individuell für jeden Empfänger automatisiert mit Daten, wie z.B. den Namen, zu füllen und diese zu versenden. Des Weiteren ist ein Email-Monitoring vorhanden, das die eigenen geschriebenen und empfangenen Emails verwaltet. Darüber hinaus werden die vom Dienst automatisch versendeten Emails gespeichert und sind für den PC Chair einsehbar und notfalls erneut versendbar.

Evaluation

Die *EvaluationMatrix* ist die zentrale Seite des Features während der Entscheidungsfindung, die für das Komitee einsehbar ist. Gutachter haben einen Lesezugriff auf die Informationen, wobei die Konferenzleiter Modifikationen der angezeigten Daten vornehmen können. Die relevanten Daten wie Artikeldaten, die Benotungen der Gutachter, der gewichtete Mittelwert der Benotungen und die daraus resultierende Rangliste der Artikel, werden mittels des Evaluation-Features berechnet und visualisiert. Die Artikelforen können zur Diskussion geöffnet bzw. geschlossen und die Benotungen vom PC Chair über das Feature modifiziert werden. Anhand der berechneten Varianz, die die Übereinstimmung bzw. Abweichung der Benotungen anzeigt, ist der Diskussionsbedarf unter den Gutachtern für eine Entscheidungsfindung zu erkennen. Der PC Chair setzt anhand der Diskussionsergebnisse den entsprechenden Artikelstatus und stellt dadurch den aktuellen Entscheidungsstatus dar, der farblich visualisiert hervorgehoben wird. Während der Diskussion kann der Zustand des Artikels zwischen mehreren Status wechseln (siehe Abbildung 5.2). Sobald ein Artikel den Status *accepted* oder *rejected* aufweist, kann der Artikel mit dem Report-Feature weiterbearbeitet werden, das dem PC Chair das Schreiben eines abschliessenden Gutachtens freischaltet. Durch die zentralisierte Darstellung der benötigten Informationen und der Gruppierung der Artikel und deren grafische Hervorhebung ist der jeweilige Status der Entscheidungsfindung für jeden Artikel schnell ersichtlich.

Forum

Sind große Abweichungen der Beurteilungen eines Artikels vorhanden, besteht Diskussionsbedarf. Jeder Artikel verfügt im OCS über ein eigenes Forum dessen Zugriff automatisch gesteuert wird. Wird ein Forum vom PC Chair für einen Artikel geöffnet, erhalten nur die Gutachter Zugriff, die ein Gutachten für ihn verfasst und bereits eingereicht haben. Die Diskussionsleitung ist Aufgabe der PC Chairs, die ebenfalls Zugriff erhalten. Wird keine Übereinstimmung erzielt, kann ein PC Chair gezielt Gutachtern, die nicht direkt in der Begutachtung des Artikels involviert waren, Zugriff auf das Forum geben. Die zugrunde liegende Seite zur Diskussion enthält neben den Forenbeiträgen, die relevanten Artikeldaten zuzüglich der geschriebenen Gutachten, was einen schnellen Gesamtüberblick verschafft. Zur Vermeidung von unnötigem Zeitaufwand für das Anmelden am Dienst und das Navigieren zum relevanten Forum wurde ein Ticketsystem realisiert, das es ermöglicht, einen direkten Zugang auf ein Forum zu erhalten. Der OCS versendet dieses Ticket an die involvierten Gutachter, sobald ein neuer Beitrag im Forum existiert. Sind die lokalen Diskussionen zu jedem Artikel abgeschlossen, kann der PC Chair gezielt die Foren der Artikel, die zur Annahme vorgeschlagen sind, für das gesamte Komitee freigeben. Es gibt in der Regel mehr Artikel, die es verdient hätten auf der Konferenz vorgestellt zu werden, als die Anzahl der akzeptierten Artikel. Es wird global diskutiert, welche der potentiellen Beiträge, letztendlich angenommen werden.

History

Dieses Feature führt eine Historie über durchgeführte Aktionen auf Objekte im OCS, diese sind unter anderem Article, Role oder Report. Die Einträge der Historie lassen sich nach verschiedenen Gesichtspunkten filtern. Im Falle der Artikel besteht die Filterung aus der Artikel-ID, der Benutzer-ID und der Historie-Nachricht (*Article has been removed* oder *Role has been modified.*), die den Eintrag spezifiziert. Diese Filterkriterien können zum Zwecke der Filterung kombiniert werden. Mit der Historie kann sich ein Überblick von den durchgeführten Aktionen bzgl. ausgewählter Objekte verschafft werden.

Production

Das *Production-Feature* stellt den kompletten Geschäftsprozess für die Erstellung eines Tagungsbandes in Form einer separaten Web-Applikation bereit, die in Kapitel 7.3 beschrieben ist. Als Grundlage für den Tagungsband dienen die Daten der akzeptierten Artikel. Hierzu zählen die Endversionen, inklusive

der Sourcen und den Metadaten. Zur Erstellung eines Autorenindexes und für die Auflistung der am Entscheidungsprozess beteiligten Personen (PC Chairs, PC Members, Organisatoren etc.) werden zusätzlich Benutzerdaten benötigt. Der erstellte Tagungsband wird i.d.R. an den Springer Verlag Heidelberg zum Druck und zur Veröffentlichung weitergeleitet.

Report

Das *Report Management* bietet Geschäftsprozesse zur Verwaltung und Bearbeitung von Begutachtungen. Ein eigenes Framework (siehe Kapitel 4.4) wurde für die Erstellung individueller Begutachtungsformulare realisiert und bietet den Konferenzleitern die Web-basierte Konfiguration der auf die Konferenz abzustimmenden Formulare an. Des Weiteren bietet diese Feature, vergleichbar mit dem Article-Feature, die Einreichung, das Lesen, die Modifikation oder Löschung eines Reports an. Die Aktionen werden über das Rollen und Rechtesystem gesteuert und können feingranular zugeordnet werden. Hervorzuheben sind die *anonymous modus* und *view confidential content* SubFeatures. Diese bewirken einerseits eine Anonymität zwischen den PC Members und andererseits die kontrollierte Anzeige der als sicherheitsrelevant gekennzeichneten Elemente eines Begutachtungsformulares.

Role

Rollen geben Benutzern Rechte, Dienstfunktionalitäten zu verwenden, Objekte zu bearbeiten oder Dienstinformationen einzusehen. Sie können neu angelegt, modifiziert oder gelöscht werden. Eine Rolle besteht aus einer Auswahl von Feature und SubFeatures, die die Geschäftsprozesse der Applikation repräsentieren. Diese Feature bilden zusammen mit dem *User Management* das Rollen/Rechte Management (siehe Kapitel 4.2), das das Konzept der flexiblen Zugriffskontrolle widerspiegelt.

Setup

Die unterstützte schrittweise Konfiguration des Entscheidungssystems wird durch das Feature *Setup* verkörpert. Das Setup führt durch einen Workflow, der eine Auswahl von Features konfiguriert und diese in Bezug auf die Anforderungen einer Konferenz flexibel anpasst. Hierzu zählen das Aussehen der Dienstseiten, Logos, Zeitfristen (deadlines), Reportformulare, Anzahl der Gutachter pro Artikel, Artikelkategorien, Vorschriften zur Einreichung und das

Copyright, Formate der Artikel, Rollen, Vorregistrierung des Komitees, etc. Des Weiteren sind alle vom Dienst automatisch zu versendenden Email als Vorlagen online modifizierbar. Der OCS kann über das Setup in verschiedenen Modi betrieben werden:

- **Setup Mode**: Der Dienst steht ausschließlich den Konferenzleitern für die Konfiguration desselben zur Verfügung.
- **Demo Mode**: Das Komitee kann den Dienst mit den durchgeführten Konfigurationen testen. Dieses beeinflusst nicht den Originaldienst, da auf einer Kopie gearbeitet wird.
- **Productive Mode**: Der Dienst wird freigeschaltet, d.h. die Komiteemitglieder erhalten ihre Zugangsdaten und können ihr Profil vervollständigen. Autoren sind in der Lage, sich zu registrieren und ihre Beitäge einzureichen.

Service Status Report

Der *Status Report* beinhaltet einen Überblick über die Situation der Konferenz. Die Inhalte dieses Reports können mittels des Setup Features konfiguriert und per Email automatisch zugeschickt werden. Es sind Handlungsbedarf sowie Fortschritte durch den Report ersichtlich.

Staff

Die Mitarbeiter (Staff) bestehen aus anderen Dienstnutzern, die zu dem Personenkreis gehören, mit dem direkt zusammengearbeitet wird. Der PC Chair bzw. PC Member sind in der vordefinierten Rollenkonfiguration in der Lage, einen eigenen Staff anzulegen. Dieses hat den Vorteil, diverse Listen auf die Anzahl der Personen des eigenen Staffs zu beschränken.

Tasklist

Die *Tasklist* ist eine Liste von Aufgaben, die der jeweilige Benutzer zu erledigen hat. Der Benutzer sieht in dieser Liste die Informationen seiner Aufgaben und die als nächstes durchzuführende Aktion. Die *Tasklist* eines PC Members enthält die ihm zugewiesenen Artikel mit der Aufgabe, einen Report zu schreiben bzw. der Möglichkeit, diesen an einen SubReviewer weiterzuleiten.

User

Das *User-Feature* verwaltet die Benutzer des Dienstes. Benutzer können angelegt, editiert, gesperrt, freigeschaltet oder gelöscht werden. Des Weiteren integriert es die Verwaltung der Rollen und Kategorienzuordnung sowie die Benutzer-spezifischen Rechtezuweisungen. Die Registrierungskomponenten basieren ebenfalls auf dem im Forum-Feature verwendeten Ticketsystem. Ein Autor erhält nach erfolgreicher Selbstregistrierung ein Ticket via Email zugeschickt, mit dem er sein Benutzerkonto freischalten und bestätigen muss.

Die vorgestellten Features sind graphbasiert mittels des ABC's als Makros realisiert. Durch den modularen Ansatz konnten die Makros in verschiedenen Web-Applikationen wiederverwendet werden. Die Applikationen gehören zur Dienstfamilie des OCS, die in Kapitel 7 vorgestellt werden.

Kapitel 6

Architektur und Technologie

Dieses Kapitel stellt die zur Entwicklung des Entscheidungssystems verwendeten Konzepte und Technologien vor. Architekturen des OCS werden aufgezeigt und Zusammenhänge sowie Abhängigkeiten beschrieben.

6.1 Client-Server-Architektur des OCS

Das Entscheidungssystem OCS basiert auf einer Client-Server-Architektur, die exemplarisch in Abbildung 6.1 dargestellt wird. Der Client (Frontend) ist ein gängiger Browser, mit dem der Dienstbenutzer Anfragen (requests) generiert und an einen Webserver (Backend) zur Verarbeitung übergibt. Als Webserver wird für den OCS der *Apache HTTP Server* [Fou09b] eingesetzt. Der Webserver leitet diese Anfrage an einen Applikationsserver weiter, der unter Verwendung von weiteren verteilten Systemen (z.B. Datenbank, Dateisystem, SAN [1], Versionsverwaltungssystem, ...) die benötigten Daten zusammenstellt, um anschließend dynamisch die angeforderte Web-Seite (response) zu generieren.

Der OCS basiert auf der Java-Servlet-Technologie. Ein Servlet ist eine serverseitige Java-Anwendung, die in einem Servlet-Container *Apache Tomcat*[Fou09c] ausgeführt wird. Hierzu muss eine Java-Klasse implementiert werden, die von der Klasse `javax.servlet.http.HttpServlet` ableitet. Im OCS wurde die Initialisierungsmethode `init()` überschrieben, um z.B. die Geschäftsprozesse, Datenbankadministratoren, die EWIS-Laufzeitumgebung (EWIS, siehe Kapitel 3.2) zu initialisieren. Des Weiteren wurde die Methode `destroy()` überschrieben, um ein gesichertes Beenden der existierenden Verbindungen zu an-

[1] Storage Area Network

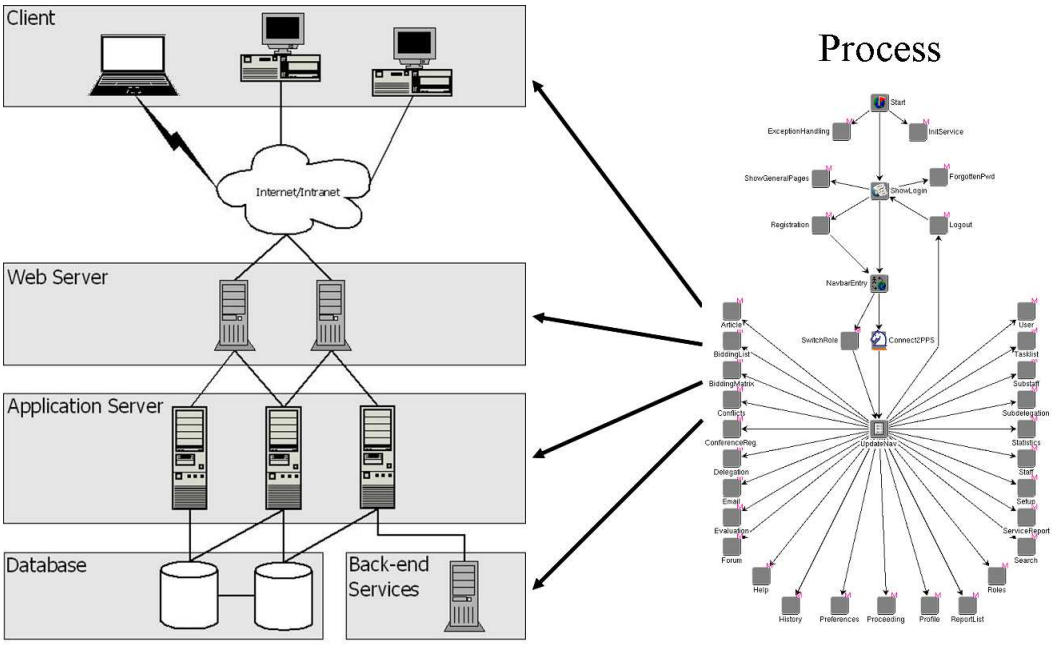


Abbildung 6.1: Client-Server-Architektur des OCS

deren Anwendungen, wie z.B. der Datenbank, zu gewährleisten und Datenverlust zu vermeiden. Um eingehende Daten bearbeiten zu können, ist mindestens eine der Methoden `doGet`, `doPOST` oder `service` zu überschreiben. Um das Servlet ausführen zu können, benötigt der Servlet-Container den *Deployment Descriptor*, eine XML-Datei namens web.xml, die Konfigurationen und Parameter für das entsprechende Servlet beinhaltet. Um die Anwendung starten zu können, werden alle benötigten kompilierten Klassen, insbesondere die Servlet-Klasse, HTML-Seiten, Bilder, etc. in ein Archiv (.war) zusammengefügt und dem *Apache Tomcat* übergeben. Das Starten und Beenden kann entweder über die Kommandozeile oder eine Web-Schnittstelle erfolgen.

Wie in der Abbildung 6.1 angedeutet, sind die einzelnen Geschäftsprozesse mittels des *ABC* als Graphen modelliert. Die Prozesse definieren den Kontrollfluss des Dienstes, der sich auf die unterschiedlichen Kontexte auswirkt. So ist ein Prozess zuständig für das Lesen aus der Datenbank, ein anderer schreibt die Daten in die Datenbank, wiederum ist ein anderer Prozess für die Änderung einer Rolle zuständig. Koordiniert werden diese Prozesse (hier Makros) von einer übergeordneten Schicht, die im Fall des OCS ebenfalls als Graph modelliert ist. Diese Struktur und die genaue Vorgehensweise ist in Kapitel 4.1.2 beschrieben. Die Erstellung eines Gesamtsystems aus zugrunde liegenden Geschäftsprozessen folgt dem SOA Konzept, das eine schnelle, flexible und einfache Änderung

von Geschäftsabläufen als Ziel hat.

6.2 Schichtenmodell des OCS

Der OCS ist ein verteiltes System, dessen Services sich auf mehreren Servern verteilen, und sich dem Benutzer als ein ganzheitliches System präsentiert. Die Verwendung eines solchen Konzepts unter der Berücksichtigung redundanter Ausführungen wichtiger Komponenten führt zu einer

- hohen Leistungssteigerung,
- besseren Skalierbarkeit und
- sehr hohen Ausfallsicherheit (24x7).

Der OCS wurde als eine fünf schichtige Web-Applikation entwickelt, dessen Aufbau in Abbildung 6.2 dargestellt wird.

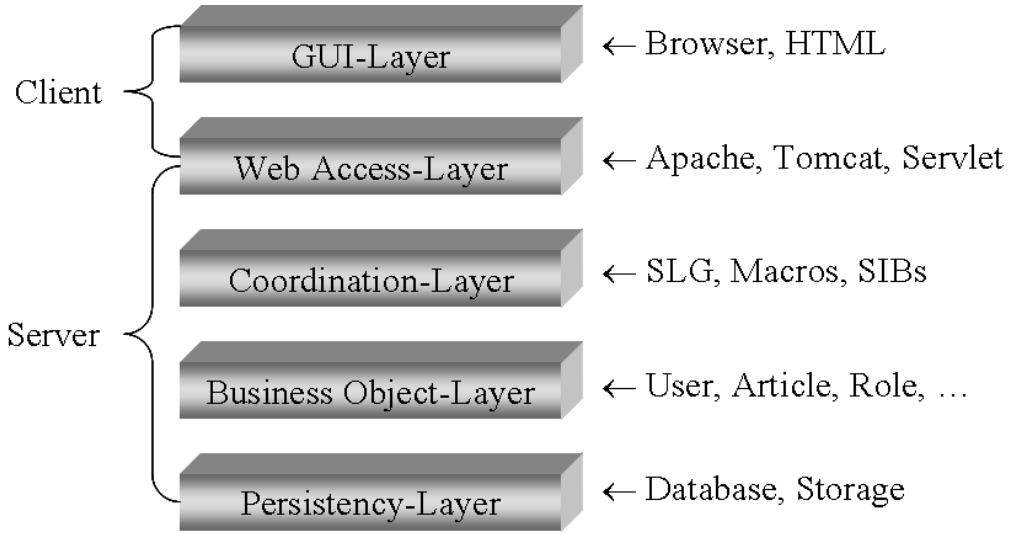


Abbildung 6.2: Schichtenmodell des OCS

Die Schichten lassen sich, wie in der Abbildung dargestellt, in Server-seitige und Client-seitige Schichten gruppieren. Der *Web Access-Layer* repräsentiert die Schnittstelle zwischen beiden Gruppen und steuert deren Kommunikation. Im Folgenden werden die einzelnen Schichten näher beschrieben:

1. **Persistency-Layer:** Die Persistenzschicht bildet die unterste Schicht. Sie ist zuständig für die Speicherung der Geschäftsobjekte in einer Datenbank sowie für die Datenhaltung in einem Versionierungssystem oder Dateisystem. Für die Datenbankanbindung wurde JDBC [2] [mic09j] verwendet, eine Java-basierte Schnittstelle für relationale Datenbanken. Mit JDBC wird eine Anbindung an Datenbanken verschiedener Hersteller ermöglicht. So kann OCS sowohl auf PostgreSQL [Com09] als auch Oracle [Ora09]-Datenbanken arbeiten.

2. **Business Object-Layer:** Diese Schicht beinhaltet die Geschäftsobjekte. In dem Projekt OCS gehören hierzu unter anderem die Objekte Artikel, Benutzer, Rollen, Assoziationen zwischen den Objekten sowie Administratorklassen. Die Schicht ist losgelöst von den Applikationen, so dass die Objekte applikationsübergreifend eingesetzt werden können.

3. **Coordination-Layer:** Die Koordinationsschicht stellt den Kontrollfluss einer Applikation dar. Im ABC wird der Ablauf eines Prozesses oder einer Anwendung als Kontrollflussgraph (SLG) modelliert, der die Geschäftslogik einer Applikation repräsentiert. Die im SLG verwendeten SIBs arbeiten auf den im *Business Object-Layer* realisierten Objekten und stellen diese in Beziehung.

4. **Web Access-Layer:** Diese Schicht ist für die Kommunikation zwischen einem Dienstbenutzer (Client) und der Applikation verantwortlich. Sie setzt sich einerseits aus dem Web-Server, der die Anfragen an die Applikationsserver weiterleitet und andererseits aus der Laufzeitumgebung (EWIS), die mithilfe der Koordinationsschicht die Geschäftslogik durchläuft, zusammen. Die durch die Abarbeitung der für die Anfrage relevanten Geschäftslogik erzielte Antwort wird dem Client übertragen und entsprechend der GUI angezeigt.

5. **GUI-Layer:** Die grafische Benutzeroberfläche stellt die Kommunikationsschnittstelle zum Dienstbenutzer in Form eines Browsers dar. Es werden Dienstinformationen bzw. Formulare als HTML-Seiten angezeigt. Der Browser dient als Eingabe- und Ausgabeinstrument, das für den OCS ohne jegliche Zusatzsoftware auskommt. Das ermöglicht das weltweite Arbeiten unter Verwendung eines rudimentären Browsers.

Anhand der spezifizierten Schichten lässt sich ein organisiertes und paralleles Vorgehen bei der Entwicklung einer Applikation verfolgen. Arbeiten können

[2] Java Database Connectivity

auf unterschiedliche Experten aufgeteilt werden. So sind die *Programmierexperten* für die Umsetzung der Schicht 1, Schicht 2 und der Laufzeitumgebung (Schicht 4) zuständig. Die SIB-Experten schreiben auf Basis der Geschäftsobjekte Dienstfunktionalitäten, die als Bibliotheken dem *Anwendungsexperten* zur Verfügung stehen. *Anwendungexperten* erstellen den Kontrollflussgraphen der Anwendung, was keinerlei Programmierkenntnisse voraussetzt. Der *HTML-Designer* entwickelt die HTML-Seiten unter Berücksichtigung der Programmierschnittstelle der Objekte, die die anzuzeigenden Informationen kapseln.

Das Schichtenmodell des OCS basiert auf dem Model View Controller (MVC) Konzept, das sich aus einem Datenmodell, einer Präsentationsebene und der Programmsteuerung zusammensetzt. Die Schichten *Persistency-Layer* und *Business Object-Layer* des OCS entsprechen dabei dem Datenmodell. Die Programmsteuerung wird von dem *Coordination-Layer* und dem *Web Access-Layer* repräsentiert. Die Präsentationsebene erstreckt sich über die Schichten *Web Access-Layer* und *GUI-Layer*.

Die Einführung eines *Coordination-Layer* als Erweiterung des Ansatzes des MVC-Modells wird als *Lightweight Process Coordination approach* [MS04b] bezeichnet. Das Konzept *Aggressive model-driven development* (AMDD) [MS04a] beruht auf dem erweiterten MVC Konzept und ermöglicht Personen ohne tiefere Kenntnisse über die Diensttechnik und Programmiersprachen, Anwendungen zu entwerfen. Darauf basiert das *eXtreme Model-Driven Design* Paradigma (XMDD) [KJMS09], welches durch die erzielte Abstraktionsebene der Modellierung der Geschäftsabläufe Applikationsexperten und Geschäftsleute gleichermaßen kontinuierlich in den gesamten Lebenszyklus eines Systems involviert.

Der OCS ist in der Programmiersprache Java [INC09] geschrieben, eine objektorientierte Sprache, deren Konzept, die Plattformunabhängigkeit der implementierten Anwendungen, auf nahezu allen Rechnerplattformen erfüllt ist. Die entwickelten Java-Klassen (.java) werden mittels eines Compilers in Java-Bytecode (.class) übersetzt. Die Plattformunabhängigkeit wird durch die JVM [3] erzielt, einem Interpreter, der für die jeweilige Plattform implementiert wird und dort den Bytecode ausführt. Die Firma Sun Microsystems [mic09l] bietet die JVMs für verschiedene Kombinationen von Hardware- und Betriebssystemen an.

[3] Java Virtual Machine

Kapitel 7

Dienstfamilie

Dieses Kapitel stellt die Dienstfamilie des OCS vor. Sie besteht zum Teil aus Diensten, die aus den Konzepten und somit aus dem OCS heraus entstanden sind und unabhängig vom OCS genutzt werden. Zum anderen Teil der Dienstfamilie gehören diejenigen, die direkt mit dem OCS arbeiten. Es sind Features des OCS, die als eigene Dienste realisiert wurden. Die Abbildung 7.1 zeigt den Zusammenhang des Gesamtsystems OCS.

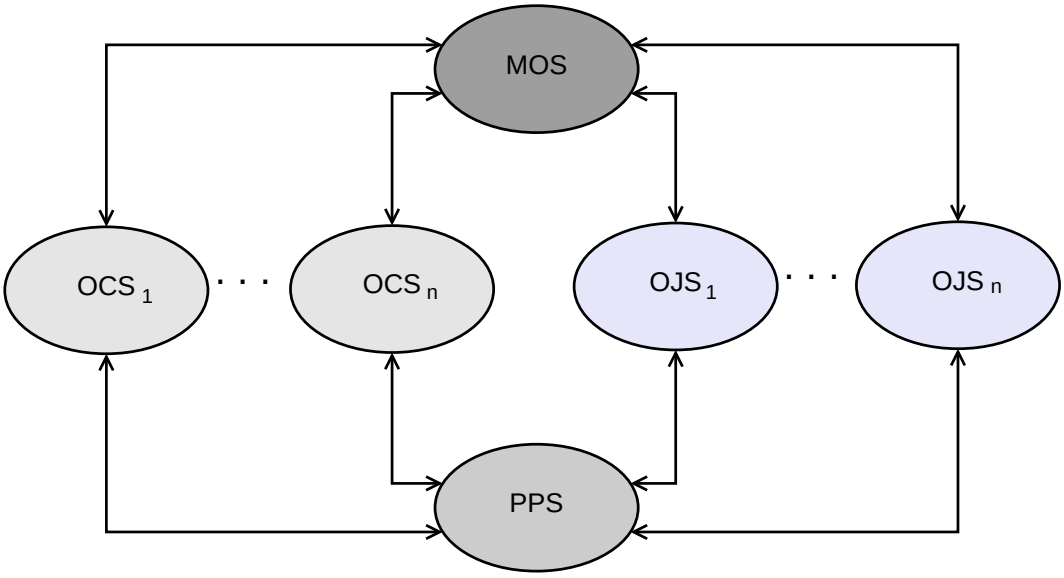


Abbildung 7.1: Architektur: MOS, OCS, OJS, PPS

Die folgenden Kapitel befassen sich jeweils mit einem Dienst der OCS-Dienstfamilie, beschreiben deren Einsatzbereich und gehen kurz auf die verwendete

Technologie ein.

7.1 Management Overview System (MOS)

Das *Management Overview System* ist ein zentraler Zugriffspunkt (Single Point of Entry) auf die Dienstinstanzen des OCS und OJS. Abbildung 7.1 zeigt den Zusammenhang auf. Der MOS gibt den OCS bzw. OJS-Benutzern (PC Chair, PC Member, Author, etc.) und den MOS-Administratoren einen direkten Zugriff auf die Instanzen. OCS, OJS-Benutzer erhalten nach erfolgreicher Authentifizierung eine Tabelle von OCS-Instanzen, in denen der Benutzer involviert ist, d.h. über einen Account verfügt. Von dieser Seite kann direkt auf die gelisteten Instanzen zugegriffen werden, ohne eine erneute Authentifizierung durchführen zu müssen. Das Ausloggen aus der jeweiligen Instanz führt automatisch zurück zur Übersichtsliste der Konferenzen. Des Weiteren gibt der MOS dem Administrator eine Verwaltungskomponente zur Hand, die überwiegend die bereits existierenden Komponenten einer OCS-Instanz direkt verwendet. Die Verwaltungsfunktionalitäten sind mit Java Server Pages (JSP) realisiert.

Ein weiterer wesentlicher Bestandteil ist die Synchronisierung von Daten. MOS wurde als eigene Web-Anwendung konzipiert, die auf einer eigenen Datenbank basiert. Für die Persistenzschicht wird das Framework Hibernate [BHRS07] eingesetzt. Änderungen, die an Objekten im MOS oder in einer OCS-Instanz durchgeführt werden, sind zwischen dem MOS und den OCS-Instanzen zu synchronisieren. Der Abgleich der Objekte findet mittels der OpenSource Lösung JMS [1] [mic09k] statt. Als JMS-Provider wird Apache ActiveMQ [Fou09a] eingesetzt. Die Kommunikation zwischen dem MOS und den Dienstinstanzen erfolgt ausschließlich über JMS und ist asynchron. Hierzu erhält jede Dienstinstanz und der MOS eine eigene Queue. Im Falle einer Änderung im OCS schreibt dieser eine Message in die Queue des MOS, der diese konsumiert. In die Queue der OCS-Instanzen darf nur der MOS schreiben, der auf diesem Wege Änderungen an die OCS-Instanzen propagieren kann. Sollte eine OCS-Instanz oder MOS selbst nicht online sein, so werden die Messages der entsprechenden Queue persistent gehalten. Sobald der Dienst wieder online ist, meldet sich dieser erneut an dem MOS an und konsumiert die in seiner Queue befindenden Nachrichten vom JMS-Provider.

Diese Lösung des übergeordneten Dienstes ist unter dem Aspekt der Verwendung der zugrunde liegenden Architektur der OCS bzw. OJS-Instanzen ent-

[1] Java Messaging Service

standen. In Bezug auf die Arbeitsweise und Kernfunktionalitäten sollten keine Veränderungen erfolgen. Dadurch ist gewährleistet, dass die Instanzen weiterhin autark fungieren können, mit der Option, diese in den MOS aufzunehmen. D.h. Instanzen die bei verschiedenen Kunden installiert sind, können in den MOS eingebunden werden. Die Anbindung an einen MOS erfolgt über die Konfiguration des jeweiligen *Deployment Descriptors* (web.xml Datei) einer OCS bzw. OJS-Instanz.

7.2 Online Journal Service (OJS)

Der OJS ist ein Web-basiertes Entscheidungssystem zur Begutachtung und Auslese von Artikeln, die in einer wissenschaftlichen Zeitschrift (Journal) veröffentlicht werden. Journal-Veröffentlichungen sind aus wissenschaftlicher Sicht von hohem Wert, allerdings im selben Maße oft schwierig zu verwirklichen. Einer der Gründe ist, dass die Journals nur eine begrenzte Anzahl von Ausgaben im Jahr mit einer geringen Anzahl von Beiträgen aufweisen. Eine andere Einschränkung kann sein, dass es für das Themengebiet eines Autors im Vergleich zu der Anzahl der Konferenzen sehr wenige Journals gibt.

Im Vergleich zum OCS, dessen Instanz im Schnitt drei bis vier Monate pro Konferenz aktiv ist, dauert die Entscheidungsfindung über Annahme oder Ablehnung eines Journal-Beitrages mitunter bis zu einem Jahr und länger. Einer der Gründe ist die Anzahl der Revisionen des Beitrages, die ein Autor aufgrund von Gutachten durchführt. In Abhängigkeit von dem Bekanntheitsgrad des Journals ist ein Begutachtungsdurchlauf als strenger einzuordnen im Vergleich zu einer Konferenz. Das soll allerdings die Bedeutung von Konferenzen nicht schmälern. So kann es vorkommen, das auch nach mehrmaliger Verbesserung und Einreichung der überarbeiteten Version, der Artikel letztendlich für das Journal abgelehnt wird.

Der OJS ist auf der Basis der OCS-Technologie realisiert worden, d.h. er ist ebenfalls mittels des ABC und unter Verwendung der EWIS-SIB-Bibliotheken als eine Web-basierte Anwendung modelliert und als Java-Servlet realisiert worden. Der Feature-basierte Ansatz wurde umgesetzt und darauf aufbauend das Benutzer/Rollen-System übernommen. Der OJS verfügt über den gesamten Umfang an Features, die für den OCS implementiert wurden. Unterschied zum OCS ist einerseits der Verzicht auf Features, wie die Diskussionforen oder das Bidding mit automatischer Verteilung der Begutachtungsaufgaben, andererseits ein abgewandelter und erweiterter Workflow durch neue Features.

Der Workflow startet mit der Einreichung der Beiträge durch die Autoren.

Das Editorial Office (Leitung der Konferenz) kann diese direkt zur Begutachtung an die Reviewer (Gutachter von Beiträgen) weiterleiten oder sich erst Vorschläge für Reviewer von Editors (Herausgeber) einholen. Ein Editor kennt die Wissensgemeinschaft und kann entsprechend Mitglieder als Reviewer vorschlagen. Das Vorschlagen der Reviewer und die Übernahme der selbigen zur Verteilung der Begutachtungsaufgaben erfolgt über das Feature *Nomination*. Die Reviewer begutachten die delegierten Artikel und schreiben ein Gutachten. Nachdem alle Gutachten eingegangen sind, entscheiden die Editors anhand der Gutachten, welche Artikel zur Veröffentlichung akzeptiert werden sollen. Die akzeptierten Artikel werden unterschieden in direkt akzeptierte Artikel und in vielleicht akzeptierte Artikel. Im Falle eines vielleicht akzeptierten Artikels durchläuft die vom Autor eingereichte, verbesserte Version des Artikels (revised version) den Begutachtungsprozess erneut. Ein Durchlauf des Begutachtungsprozesses endet mit dem Versenden von abschließenden Gesamtgutachten an die Autoren.

7.3 Proceeding Production Service (PPS)

Nach Abschluss der Entscheidungsfindung und der Einreichung der letzten überarbeiteten Version der akzeptierten Artikel, liegen die Quelldaten (z.B. in LaTeX) und eine druckbare Version (z.B als PDF) von jedem Artikel vor. In diesem Konferenzstatus beginnt die Proceedings-Phase (siehe Abbildung 5.1). Es ist für den Workshop oder die Konferenz ein Tagungsband zu erstellen, der von einem Verlag gedruckt und veröffentlicht wird. Derartige Tagungsbände dienen einerseits als Übersichtsmaterial für die an der Tagung teilnehmenden Personen, andererseits stellt sie der Verlag zum Kauf zur Verfügung. Interessenten sind meist Wissenschaftler, die in dem selben Themengebiet forschen und der industrielle Bereich, der den neusten Stand der Forschung verfolgt um diesen eventuell gewinnbringend im eigenen Unternehmen einfließen zu lassen.

Die folgenden Elemente bilden einen Tagungsband:

- Der Titel des Tagungsbandes.
- Die Reihennummer des Bandes (volume number).
- Die Einleitung und Danksagung.
- Die angenommenen Artikel.
- Auflistungen aller Gutachter und der Organisatoren.

- Das Inhaltsverzeichnis.
- Der Autorenindex.

Der PPS wurde realisiert, um den Organisator der Konferenz bei der Erstellung des Tagungsbandes zu unterstützen. PPS liest die relevanten Daten von der OCS- oder OJS-Instanz ein und verwaltet diese in der eigenen Datenbank. Die Web-Applikation führt den Anwender durch die relevanten Schritte zur Anfertigung des Tagungsbandes. Die Gliederung des Bandes erfolgt durch die Einordnung der Artikel in thematische Gebiete, wobei der Titel eines Artikels und die Namen der Autoren die Beschriftung eines Gliederungspunktes sind. Diese Einteilung der Artikel wird anschließend durch das Einfügen von Überschriften (Kapiteln) verfeinert. Die grafische Repräsentation fügt dafür ein neues Textfeld ein, in dem die Überschrift eingegeben wird. PPS bietet jederzeit die Möglichkeit, die Proceedings als PDF-Vorschau zu generieren. Dadurch sind auf einfache Weise die Struktur und eventuelle Fehler in der Formatierung zu erkennen. Zusätzlich können Latex-Sourcen der Artikel mittels der Dokumentenklasse des Verlages auf ihre Gültigkeit hin überprüft werden. Sobald die Erstellung des Tagungsbandes abgeschlossen ist, kann vom PPS ein Archiv mit allen benötigten Daten für den Verlag erstellt werden, wobei das Archiv einem durch den Verlag vorgeschriebenen Format entspricht.

Die Kernkomponenten des PPS wurden graphbasiert im jABC modelliert und mit dessen Code-Generator Plugin in Java-Klassen konvertiert, die eine Bibliothek zur Tagungsbanderstellung darstellen. Die Ausführung der Kernkomponenten wurde mittels des GWT [2] [Goo09] umgesetzt, ein Framework für Web-Anwendungen, das im Mai 2006 von Google veröffentlicht wurde. Für jede Kernkomponente ist ein GWT-Servlet implementiert, das die Kernkomponenten auf RPC [3]-Methoden abbildet. GWT ermöglicht die Entwicklung von AJAX [4] [All09]-Anwendungen, ein Konzept zur asynchronen Datenübertragung zwischen dem Server und dem Browser (Client). Somit können gezielt Daten einer HTML-Seite nachgeladen werden ohne diese komplett neu laden zu müssen. Mittels GWT lassen sich AJAX-Anwendungen in Java implementieren und durch einen GWT eigenen Compiler nach AJAX-fähiges JavaScript übersetzten. Als GUI werden AJAX-basierte HTML-Seiten verwendet, die über die GWT-Servlets die definierten RPC-Methoden aufrufen und somit Zugriff auf die Funktionalitäten der Kernkomponenten haben. Die Modellierung der Verbindungen der Seiten wurde mit dem jABC Plugin JEEWAB durchgeführt,

[2] Google Web Toolkit
[3] Remote Procedure Call
[4] Asynchronous JavaScript and XML

das das HTML-Gerüst und den Deployment Descriptor generiert. Die Datenbankschicht basiert auf dem Hibernate Framework.

7.4 Lecture Notes in Computer Science (LNCS)

Der LNCS Dienst wurde auf Basis der OCS-Technologie realisiert und setzt überwiegend auf dessen Features auf. Konzipiert wurde der Dienst zur Entscheidungsfindung über die Zulassung eines Projektes zur Publikation. Ein Projekt kann z.B. eine Konferenz, ein Journal oder ein Workshop sein. Ein Organisator reicht seinen Vorschlag beim Verlag zur Publikation ein.

Die Rolle *Editor in Chief* verwaltet und koordiniert den Begutachtungsprozess. Eingehende Vorschläge bzw. Anfragen werden mittels dieser Rolle über ein Formular in den Dienst eingepflegt. Es werden verschiedene Metadaten zum Projekt eingegeben, wie der Name des Ansprechpartners, der mit der Rolle *Organizer* im Dienst angelegt wird, oder die Internetseite des Projektes (Homepage der Konferenz). Diese Informationen sind für den *LNCS Series Editor* bestimmt, um sich ein Bild über den Vorschlag zu machen. Durch die Benachrichtigung der entsprechenden *LNCS Series Editor* verteilt der *Editor in Chief* die Anfragen auf Veröffentlichung. Zeitgleich wird der Status des Beitrages auf *discussion* gesetzt und automatisch ein Forum zur Diskussion geöffnet. Die Foren dienen den Reihenherausgebern zur Entscheidungsfindung über Annahme oder Ablehnung der Anfrage. Für die flexible Koordination und Verwaltung der Zuweisung des Vorschlages und der Vergabe der entsprechenden Zugriffe auf das Forum, wurde das im OCS bestehende Feature *Conflict* eingesetzt. Dieses zeigt die Flexibilität und die daraus resultierende Wiederverwendbarkeit der Features in anderen Kontexten auf. Das Ergebnis der Diskussion wird dem Organizer über den Dienst unter Verwendung von konfigurierbaren Email-Templates zugeschickt.

7.5 Transactions

Der *Transactions Dienst* gliedert sich ebenfalls in die Dienstkategorie der Entscheidungssysteme ein und verfolgt auch den Feature-orientierten Ansatz. Er beinhaltet Dienstfunktionalität des OCS und OJS und erweitert den Umfang an Features durch eigene Applikationsspezifische Features. Mittels des ABC werden die Features, die in Form von Makros vorliegen, zu einen Kon-

trollflussgraphen modelliert, der die Geschäftslogik des Transactions-Dienstes repräsentiert.

Im Transactions-Dienst ist ein *Editor in Chief* zuständig für die Konfiguration des Dienstes und dessen Leitung. Er leitet die von dem Autoren eingereichten Forschnungsergebnisse an die *Editors* (Herausgeber) weiter. Diese verfassen entweder ihr eigenes Gutachten, oder schreiben eine Zusammenfassung von den *Reviewer*-Gutachten, an die sie weiterdelegieren können. *Reviewer* sind ausschließlich für das Verfassen von Gutachten zuständig. Die Diskussion bzgl. der Annahme eines Beitrages wird vom *Editor in Chief* und den *Editors* geführt. Es sei erwähnt, dass es nicht nur einen *Editor in Chief* geben muss. Aufgrund der Diskussion kann der Beitrag in den Status *accepted*, *proposed accepted* oder *rejected* gesetzt werden. Nach dem Setzen des Artikels in einen der Zustände wird automatisch die Funktionalität für das Schreiben der Abschlussgutachten freigegeben. Das Verfassen eines Abschlussgutachtens hat in Abhängigkeit des Artikelstatus folgende Auswirkung auf den weiteren Prozess:

- *accepted*: Der Autor darf eine überarbeitete Artikelversion als Endversion seiner Arbeit einreichen.
- *proposed accepted*: Der Autor reicht die überarbeitete Version seines Artikels ein, die erneut den Begutachtungsprozess durchläuft. Die Wiederholung dieses Prozesses endet, sobald der Artikel den Zustand *accepted* oder *rejected* einnimmt.
- *rejected*: Der Artikel ist abgelehnt. Die Autoren können die Gutachten zu ihrem Artikel einsehen.

Ein weiterer wichtiger Punkt bei dem Begutachtungsprozess des Transactions-Dienstes sind die individuellen Zeitfristen. Im OCS ist eine Synchronisierung der Begutachtungsphasen für alle Artikel zwingend notwendig, da die Artikel in einem konkurrierenden Verhältnis stehen und Entscheidungsfindungen alle Artikel betreffen. Im Transactions-Dienst hingegen kann die Zeitfrist, die die Beendingung des gesamten Begutachtungsprozesses spezifiziert, entweder für einen einzelnen Artikel oder für eine Artikelkategorie festgelegt werden.

Teil IV

Analyse und Erfahrungen: OCS im Einsatz

Kapitel 8

Design-Entscheidungen

Dieses Kapitel enthält eine kurze Zusammenfassung der angewandten Technologien und diskutiert eine Auswahl von Dienstfunktionalitäten (siehe auch Kapitel 5.3), die im Laufe der Jahre das Einsatzspektrum der Entscheidungssysteme erweitert haben. Eine detailliertere Darstellung der für den OCS verwendeten Technologien kann in den Kapiteln 3 und 6 nachgeschlagen werden.

Die Geburtsstunde des OCS war 1999 in der Firma METAFrame Technologies. Als Entwicklungsumgebung wurde das firmeneigene Entwicklungswerkzeug ABC [SM99, SMCB96] (siehe Kapitel 1.2.3 und 3.1) eingesetzt. Das ABC war bereits Ende der neunziger Jahre mit Hilfe von EWIS (siehe Kapitel 3.2) in der Lage, graphbasiert Web-Applikationen zu modellieren und den Kontrollfluss auf Basis der Graphen zu generieren und auszuführen. Dabei wird ein `WAR`-Archiv der Web-Anwendung erstellt, das in einem Webcontainer (z.B. Tomcat) ausführbar ist. Das ABC repräsentiert ein einheitliches Werkzeug zur modellbasierten Entwicklung von komplexen Web-Anwendungen auf Ebene von Geschäftsabläufen [MS09, SN07, MS04a, KJMS09], dessen Modelle ausführbar und validierbar sind [KM05, KM06, KMW08]. Die Möglichkeit der modellgetriebenen Softwareentwicklung (MDA) und die daraus folgende strukturierte Vorgehensweise der Konzeption (SOA) erfüllte die an der Applikation gestellten Anforderungen, wie Modularität und Wiederverwendbarkeit sowie die Erweiterbarkeit und die einhergehende Modifizierbarkeit des Systems (siehe Kapitel 1.2.2). Die vereinfachte Form der Abbildung von komplexen Geschäftsprozessen auf die leicht verständliche graphbasierte Darstellungsform ermöglicht die Involvierung von Nicht-Programmierern in den Prozess der Übertragung der Geschäftsabläufe in die IT-Welt. Dieser wesentliche Aspekt wurde von anderen Entwicklungsumgebungen wie *LEU* [DGSZ94, Bra01] oder *MOKASSIN* [HG98, Bra01] nicht verfolgt. Das *ARIS*-Konzept [Sei02] fokus-

sierte sich durch die Verwendung der Modellierungssprache *EPK* [HW08] auf die Modellierung von Geschäftsprozessen, bot aber keine Möglichkeit für die Ausführung derselben. Des Weiteren lassen sich die Graphen des ABC mit temporallogischen Formeln auf ihre Gültigkeit überprüfen. Mit dem Model Checking lässt sich bereits zur Modellierungsphase die geplante Anwendung testen.

Steckten die Technologien und Architekturen für die Realisierung von komplexen Web-Applikationen Ende der Neunzigerjahre noch in den Kinderschuhen oder wurden diese teilweise erst später konzipiert, so hat man heute eine nahezu freie Auswahl im kommerziellen sowie im Open Source Bereich. Die Vorgehensweise der Softwareentwicklung hat sich über die Jahre gewandelt. So musste noch vor einigen Jahren aufgrund von fehlender Software-Bibliotheken und Standards vieles von Grund auf selbst entwickelt werden. Es waren immer mehr proprietäre Lösungen vorzufinden. Ein Beispiel ist die für den OCS entwicklete Datenbankschicht, die direkt auf Basis der Datenbankschnittstelle JDBC [1] [mic09j] realisiert wurde, wobei die Datenbankinteraktionen in den Geschäftsobjekten und Administratoren gekapselt wurden. Heutzutage wird dem Entwickler mit der JPA [2] [BHRS07] [mic09e] eine Lösung an die Hand geben, welche die Applikation von der Datenbankschicht entkoppelt und die Datebankkommunikation kapselt. Die Einführung von Standards durch Arbeitsgemeinschaften, wie z.B. der OMG oder W3C, und die Wiederverwendung von existierender Software erleichtert die Arbeit der Entwickler. Paradigmen wie MDA und SOA haben einen entscheidenden Anteil an der aktuellen Softwareentwicklung. Die heutige Verfügbarkeit von Architekturen, Frameworks und Sprachen wie z.B. Java EE [mic09d], .Net [Mic09b], Spring [OLW+08], JBoss Seam [JBo09], JSF [mic09g], BPEL oder LDAP [Zör08] [LU06], stellen eine große Bandbreite von direkt verwendbaren Konzepten und Technologien bereit.

Die Integration von Nicht-Programmierern und Endnutzern in die Anwendungsentwicklung ist zu einem zentralen Punkt in der heutigen Softwareentwicklung geworden. Ein Beispiel für die einfache Unterstützung von Endnutzern sind *Mashups* [ZRN08, DLHPB09]. *Mashup* bezeichnet die Zusammenstellung von existierenden Web-Inhalten oder Web-Applikationen. Hierzu dienen Mashup-Editoren wie z.B. IBM Damia [SAM+08], Microsoft Popfly [Mic09a], Yahoo Pipes [Yah09] oder Intel MashMaker [EG07], die den Benutzer während der Kombination von existierenden Web-Applikationen grafisch durch Graphmodelle oder Flussdiagramme unterstützen.

[1] Java Database Connectivity
[2] Java Persistence API

8.1 Technologische Ebene

Auf technologischer Ebene wurden im Laufe der Zeit Änderungen einerseits aufgrund von neuen oder veränderten Java-Bibliotheken, andererseits eines zu hohen Wartungsaufwandes bzw. Vereinfachung des Installationsvorganges durchgeführt. Nicht zu vergessen sind die für einen Benutzerdienst wichtigen konstruktiven Benutzervorschläge (user feedback) für Verbesserungen oder neue Features, die ebenfalls ihre Berücksichtigung fanden.

Während der Entwicklung, Wartung und Weiterentwicklung wurde darauf geachtet, dass der Dienst mit den neuesten Releases z.B. von Java, Java Servlet und Tomcat arbeitet, um auf dem neuesten Stand zu bleiben und so neue Bibliotheken verwenden zu können. Dadurch war gewährleistet, dass selbst geschriebene Komponenten durch standardisierte Bibliotheken ersetzt werden konnten, um so den eigenen Wartungsaufwand zu minimieren. Allerdings hatte die Umstellung auf neue Releases bzw. die Aktualisierungen oder der Wechsel des Betriebsystems zufolge, dass die vom Dienst genutzte Drittanbietersoftware und die eigens dafür implementierten Schnittstellen Anpassungen unterzogen werden mussten. Betroffen waren z.B. die Speicherung der Dateien des Artikels im Versionsverwaltungssystem CVS [3] oder die Diskussionskomponente auf Basis eines Newsservers (INN2 [4]). Der Rückbau dieser Komponenten führte zur Verringerung des Wartungsaufwandes und in erster Linie zur erheblichen Vereinfachung der Dienstinstallation.

Ein Beispiel für die Veränderung der Technologie ist die Ersetzung von eigenen Implementierungen durch neue Bibliotheken, wie die implementierte Komponente für das Lesen einer hochgeladenen Datei. Ende der neunziger Jahre gab es noch keine freie Bibliothek, die `multipart` Anfragen gänzlich verarbeiten konnte. Diese Art von Anfragen übertragen neben den einfachen HTML-Elementen, Dateien, die der Dienstbenutzer per Browser mitschickt. Die eigene Implementierung wurde im Hinblick auf die weitere Kompatibilität durch die später erschienene Bibliothek `Apache Commons FileUpload` ersetzt. Bei Neuentwicklungen von Diensten des OCS wurden ebenso neue Technologien wie z.B. Hibernate, GWT oder JSP eingesetzt.

Die komplexeren Features wie die Tagungsbandkomponente (PPS) (siehe Kapitel 7.3) oder das Verwaltungsmanagement (MOS) (siehe Kapitel 7.1) mit zentralem Zugriffspunkt (Single Point of Entry) auf die OCS- und OJS-Instanzen, wurden als eigene Web-Anwendungen entwickelt. Für die Entwicklung wurden verschiedene Vorgehensweisen verfolgt um die geeignetsten Technologien für

[3] Concurrent Versions System
[4] InterNetNews

den OCS zu evaluieren.

8.2 Feature Ebene

Aus der Sicht der Funktionalitäten wurde der Dienst im Laufe der Zeit durch neue und geänderte Features erweitert bzw. modifiziert.

Um die Support-Leistungen der Konferenzleitung zu minimieren wurde neben einer FAQ-Seite und einer Willkommensseite mit integrierter Workflow-Anzeige, ein Help-Feature umgesetzt. Dieses Feature ist personalisiert in dem Sinne, dass es dem Benutzer eine Hilfefunktion in Abhängigkeit seiner Rechte zeigt und für den Benutzer eine individuelle Hilfe anbietet.

Ein mitunter simples und heutzutage gängiges Feature war die frühe Einführung einer Komponente zur automatischen Einholung eines neuen Passwortes. Die Konferenzleitung hatte zuvor einen erheblichen Support-Aufwand durch das Vergessen von Passwörtern.

Die Einführung einer Ticketkomponente unterstützt die Diskussion im OCS, durch Benachrichtigung aller, in einem Forum involvierten, Benutzer. Dieses beschleunigt den Diskussionsfluss, da die Diskussionsbeteiligten direkt per Email über den Eingang einer neuen Nachricht erfahren und sie mittels des mitgeschickten Tickets einen direkten Zugriff auf das relevante Forum erhalten. Dadurch entfällt das wiederholte Anmelden am Dienst. Darüber hinaus wird das Ticketsystem für die eigenständige Registrierung eingesetzt. Autoren registrieren sich unter Zuhilfenahme der sich im öffentlichen Bereich lokalisierten Registrierungsseite. Um Missbrauch zu vermeiden erhält der Benutzer ein Ticket per Email, mit dem die Registrierung bestätigt werden muss.

Das Bidding-Feature, das im Kapitel 4.3 ausführlich beschrieben wurde, war, wie die Erfahrung bestätigt hat, eine wichtige Design-Entscheidung, die eine erhebliche Arbeitsentlastung für die Konferenzleitung darstellt.

Um die Einbeziehung der Co-Autoren und der SubReviewer zu ermöglichen, wurden die Grundfunktionalitäten erweitert. Co-Autoren sind dadurch von Beginn an im Workflow integriert und können eingreifen, falls der Autor verhindert ist. Ein weiterer Vorteil sind die dadurch bereits im Dienst existierenden Daten der Co-Autoren, die für die Erstellung des Tagungsbandes benötigt werden.

Durch die Umsetzung des *SubDelegation* Features wurden die Weiterdelegierungen der Begutachtungsaufgaben von PC Membern an SubReviewer mit in den Konferenzprozess aufgenommen. Sub-Delegierungen müssen nicht mehr

kompliziert am Dienst vorbei geschehen. Die SubReviewer sind somit bereits im Dienst registriert und deren Daten können ebenfalls ohne größeren Aufwand für den Tagungsband verwendet werden.

Eine zusätzliche Flexibilität hat der OCS durch die Erweiterung des Benutzer/ Rollen-Rechtemodells (siehe Kapitel 4.2.1) mit dem personalisierten Rechtemanagement (siehe Kapitel 4.2.3), sowie durch die konfigurierbaren Begutachtungsformulare (siehe Kapitel 4.4) und die konfigurierbaren Foren erhalten. Durch die Konfigurierbarkeit dieser und weiteren Dienstfunktionalitäten ist es möglich, das Entscheidungssystem an die unterschiedlichsten Anforderungen online anzupassen.

Grundlegende Änderungen am Kontrollfluss können graphbasiert durchgeführt werden, wobei das umgesetzte Feature-basierte Konzept durch seine Strukturierung und Klassifizierung der Features zu einer Vereinfachung beiträgt. Mit dem entwickelten *Service Status Report* Feature (siehe Kapitel 5.3) und dem *History* Feature (siehe Kapitel 5.3) wurde der Konferenzleitung gezielt ein Monitoring-Werkzeug an die Hand gegeben, um den Verlauf der Konferenz besser nachvollziehen zu können.

Kapitel 9

Einsatz der Entscheidungssysteme

Die in dieser Arbeit beschriebenen Entscheidungssysteme finden ihre Verwendung im wissenschaftlichen Bereich. Primäre Aufgabe ist es, mit definierten Prozessen bei Entscheidungsfindungen zielgerecht in Form einer Web-Anwendung alle beteiligten Personen zu unterstützen. Es ist und soll nicht die Aufgabe eines Entscheidungssystems sein, den Verantwortlichen die Entscheidungen abzunehmen. Es schlägt lediglich eine eventuelle Konstellation in Form einer Rangliste vor, die auf Basis der Benotungen, der verfassten Gutachten errechnet wird. Die Entscheidungen treffen in letzter Instanz die Verantwortlichen selbst. Dieses ist im Bereich der Konferenzen das Programmkomitee, das sich aus den *PC Chairs* (Konferenzleitung) und den *PC Members* (Gutachtern) zusammensetzt.

Im Folgenden wird auf die Einsatzbereiche der beschriebenen Entscheidungssysteme OCS, OJS, Transactions und LNCS eingegangen. Deren Aufgaben umfassen die Vorbereitung von Workshops, Konferenzen und Fachzeitschriften (Journals) sowie die eigentliche Entscheidungsfindung eines Verlages, die Ergebnisse in Form eines Buches oder einer Fachzeitschrift zu veröffentlichen. So genügt z.B. nicht jede Konferenz den Ansprüchen zur Veröffentlichung, die ein Verlag intern festlegt. Hierzu gehören unter anderem die fachliche Ausrichtung und die Qualität der Veranstaltung aus denen die Artikel hervorgehen. Der wirtschaftliche Aspekt, der sich durch das Publizieren von qualitativ hochwertigen Konferenzen oder Journals durch die Verkaufszahlen widerspiegelt, ist ebenfalls nicht zu vernachlässigen.

Der OCS, Transactions und LNCS werden auf einem Springer-eigenen Server-

Cluster von Springer SBM [1] in Dordrecht gehostet. Der OCS wird für Konferenzen angeboten [Hei09b], die zur Publikation beim Springer Verlag Heidelberg angenommen sind. Für die Entscheidungsfindung, wer bei Springer LNCS [Hei09d] oder LNBIP [2] [Hei09d] publiziert, wird der LNCS-Dienst (siehe Kapitel 7.4) eingesetzt. Konferenzen, Workshops, etc. stellen Anfragen, die mittels des Dienstes diskutiert werden. Der Transactions Dienst wiederum wird vom Springer Verlag für die *LNCS Transactions Subline* [Hei09a] eingesetzt. Transactions sind vergleichbar mit Journals, mit dem Unterschied, dass Transactions keine vorgeschriebenen Veröffentlichungsintervalle aufzeigen und der Workflow differiert.

Des Weiteren existiert ein Referenz-Cluster des von SBM Springer in Betrieb genommenen Hochverfügbarkeitsclusters (HA-Cluster) des OCS am Lehrstuhl für Programmiersysteme der TU Dortmund. Dieses dient zur Entwicklung und für die Testphasen von Release-Kandidaten, bevor diese auf das Live-Cluster gespielt werden. Am Lehrstuhl selbst werden OCS- sowie OJS-Dienste gehostet, die für Konferenzen und Journals eingesetzt werden, wodurch ein geplantes Release einem Test unter realen Bedingungen unterzogen werden kann.

Das OCS Server-Cluster ist als Hochverfügbarkeitscluster (HA-Cluster) konzipiert worden, um eine maximale Ausfallsicherheit zu gewährleisten. Dieses wurde mitunter durch Redundanz von Hardware und Software erreicht. Switches, Verkabelungen und die Server für die Datenbanken, den Webzugriff und das Filesystem wurden redundant ausgelegt. Daten der Datenbanken oder des Filesystems werden auf einem SAN Storage abgelegt, das gesichert ist. Die Fileserver werden von einer Clustersoftware verwaltet und die Datenbanken repliziert. Beim Ausfall eines Applikationsserver, auf denen die Dienstinstanzen laufen, wird automatisch ein sich im Standby befindender Server aktiviert, der die Aufgaben des ausgefallenen Servers übernimmt. Für die Administration, das Monitoring und die Hochverfügbarkeit wird das System MaTRICS [3] [BM08, BM06, BMS06] eingesetzt, das ebenfalls auf das modellgetriebene Konzept basiert.

Die Entscheidungssysteme wurden bislang für 168 internationale Veranstaltungen mit insgesamt über 30.000 Benutzern eingesetzt. Der größte Anteil ist dem OCS zuzuordnen, der die Vorbereitung und ggf. Nachbearbeitung von Konferenzen unterstützt. Die Journals, die teilweise auf einer Auslese von angenommen Artikel der ausgewählten Konferenzen aufbauen, werden hingegen mittels des OJS umgesetzt. Die Artikel werden mit anderen auserlesenen Arti-

[1] Springer Science+Business Media, Dordrecht, Niederlande
[2] Lecture Notes in Business Information
[3] Management Tool for Remote Intelligent Configuration of Systems

keln nochmals diskutiert, um diese in einem Journal (special issue) zu veröffentlichen. Verwaltete Konferenzen sind z.B. TACAS [4] eine in der Wissenschaftsgemeinschaft bekannte Veranstaltung der ETAPS [5], ATPN [6], oder ADMA [7], eine in China stattfindende Konferenz. Eine Übersicht der bislang durch die Dienstfamilie unterstützten Veranstaltungen ist in Tabelle 9.1 zusammengestellt.

[4]Tools and Algorithms for the Construction and Analysis of Systems
[5]European Joint Conferences on Theory and Practice of Software
[6]International Conference on Application and Theory of Petri Nets
[7]International Conference on Advanced Data Mining And Applications

Jahr	Konferenzen - Journals - Workshops
2000	TACAS
2001	CHARME, EMODELS, SoftwareTrends, SPIN, STTT, TACAS
2002	CAV, ESOP, FASE, FMICS, IDPT, SPIN, STTT
2003	CC, CHARME, FASE, FME, FOSSACS, STTT, STTT-CHARME, STTT-FASE, TACAS
2004	AVOCS, CC, CPN, ESOP, FASE, FOSSACS, ICECCS, ICWE, ISOLA, SMTT, SPIN, STTT, STTT-FASE, STTT-FML, STTT-SMTT, STTT-VMCAI, TACAS, TOOLTACAS, STTT-TACAS, VMCAI
2005	ATPN, CALCO, CC, DARPA, ESOP, FASE, FM, FOSSACS, FMICS, FMICS-HB, ISOLA, LNCS-ISOLA, Perf-QUANT-STTT-ISOLA, SPIN, STTT, STTT-CPN, STTT-FASE, STTT-FMICS, STTT-ISOLA, TACAS, TCS-ISOLA, TOOLTACAS, TSDPS, WRAC
2006	ATAV, ATPN, CC, ESOP, FASE, FM, FMICS, FOSSACS, FRCSS, ISOLA, ISOLA-TRACKS, LNCS, LNBIP, MDA, PEPM, SEW, SC SPIN, STTT, STTT-SFB614, STTT-SPIN, TACAS, TGC, WRAC
2007	ATPN, CALCO, CS, DASC, DHMS, FASE, FMICS, FOSSACS, ISOLA, JODS, PLOP, ROUGHSETS, SEFM, SEW, SPIN, STTT, STTT-FMICS, STTT-HVC, STTT-SPIN, STTT-TACAS, STTT-T3UC, STTT-WQVV, SWSC-BOOK, TACAS, TOE, TOPNOC
2008	AAIM, ATPN, BICC, EUROHAPTICS, FASE, FMICS, ICECCS, ICSOC, ISOLA, PCM, SAFECERT, SEW, SSSPR, STTT, STTT-GRABATS, STTT-TTCN, TACAS, WASA
2009	AdHocNow, ADMA, ALT, COCOA, DPM-SETOP, DS, EUROSSC, HAID, ICA, ICONIP, ICSR, ISAAC, ISCLS, ISOLA-BIO, ISOLA-MED, IWSOS, PN, PReMI, ProvSec, PRIMA, RSFDGrC, SAFECERT, SAGA, SEFM-NGN, SMTEB, STTT, STTT-VSTTE, STTT-WSE, TACAS, TAMODIA, ...

Tabelle 9.1: Einsatz der Dienstfamilie - Veranstaltungen

Kapitel 10

Fallstudie: Verwendung des OCS

Ein System, das Tausenden von Benutzern zur Verfügung steht und diese bei ihrer kooperativen Zusammenarbeit unterstützt `lebt` von der Zufriedenheit der Benutzer. Die Rückmeldungen (feedback) der Benutzer nehmen, neben den eigenen Erfahrungen, eine zentrale Stellung in der Kunden-orientierten Entwicklung von Systemen ein. Um einen genaueren Einblick in die Benutzung eines Systems zu erlangen, insbesondere der einzelnen Dienstfunktionalitäten (Features), reichen diese Informationen mit unter nicht aus. Sollen Aussagen explizit über die Nutzung der Features gemacht werden bedarf es einer genaueren Untersuchung. Es lassen sich durch die Analyse Rückschlüsse auf die zentralen Features und die eventuelle Notwendigkeit der Modifikation derselbigen oder des Workflows ziehen. Dieses kann der Falls sein, wenn ein Feature durch einen sehr hohen Benutzungsgrad, die ihm untergeordneten Features blockiert und sich somit als Engpass herausstellt.

Die Betrachtung der einzelnen Features ist insofern wichtig, da diese die Palette von wiederverwendbaren Komponenten repräsentieren. Eine Verbesserung eines Features lässt sich somit auch in allen Diensten, in denen es verwendet wurde, auf einfache Weise übernehmen bzw. in neuen Diensten durch den Feature-orientierten Ansatz leicht hinzufügen.

Ein weiterer wichtiger Aspekt ist der Vergleich von verschiedenen Konferenzen im Bezug auf die Verwendung der Features und somit der Workflows. Dieses gibt einen Einblick in die Allgemeingültigkeit von Workflows. Diese Art von Studie erzeugt Fakten über die Einführung neuer oder geänderter Features. So lassen sich Design-Entscheidungen durch den Vergleich von Fallstudien auf ihren Erfolg hin überprüfen. Des Weiteren zeigt die Studie Arbeitsprofile, was für den Aspekt des Supports entscheidend ist.

Dieses Kapitel dient einerseits zur Analyse der Nutzung der Features und andererseits zur Untersuchung des Systems in Bezug auf die Ergonomie. Es wird eine aktuelle Studie vorgestellt, die auf die im Jahr 2002 durchgeführte und publizierte [MK02] Bezug nimmt.

10.1 Hintergrund

Betrachtet wird die Verwendung der Features am Anwendungsfallbeispiel OCS. Die Ergebnisse von drei Konferenzen werden untereinander verglichen und der früheren Studie gegenübergestellt. Dabei sollen Kenntnisse über veränderte Workflows und bestehende bzw. neue Features erlangt werden.

Zur Laufzeit der Konferenzen werden Aktivitäten des Dienstes in ein festgelegtes Format gespeichert. Dieses sogenannte Logfile enthält verschiedene Arten von Einträgen, die dem nachstehenden Muster entsprechen:

```
<date and time><logtype><operation description><operation id>
```

Dabei steht

- `<date and time>` für den Ausführungszeitpunkt der Operation,
- `<logtype>` spezifiziert den Typ des Protokollierungseintrages,
- `<operation description>` beschreibt die durchgeführte Operation und
- `<operation id>` kennzeichnet die Operation.

Der nachstehende Logfile-Eintrag weist einen Zugriff auf das User Features des OCS auf.

```
25.06.09 15:59 DEBUG executing code of SIB ShowUserList
```

Der Eintrag ist vom Typ `DEBUG` und hält die Ausführung eines SIB fest, der für die Darstellung der Benutzerliste zuständig ist. Diese Art von Informationen bilden die Daten für die Untersuchung des OCS.

Neben dem Typ DEBUG existierenden noch eine Reihe von Log-Einträgen, die allerdings nicht für diese Fallstudie relevant sind. Aufgrund dessen werden die Rohdaten in zwei Schritte für die eigentliche Auswertung aufbereitet.

Im ersten Schritt werden alle relevanten Log-Einträge des Typs `DEBUG` extrahiert um aus deren Informationen die stündlichen, wöchentlichen und gesamten

Aufrufe eines jeden SIBs zu berechnen. Im zweiten Schritt werden die bereits berechneten Daten ausgewertet. SIBs werden einem Feature zugeordnet und im CSV [1]-Format wie folgt gespeichert:

```
Weekly,Mo,Di,Mi,Do,Fr,Sa,So,Total
Feature articles,823,628,559,1240,1631,426,695,6002
Feature bidding,45,46,62,87,29,38,35,342
Feature forum,1341,1154,1014,1354,437,216,464,5980
```

In dem Beispiel sind die wöchentlichen Zugriffe auf die Features *Articles*, *Bidding* und *Forum* aufgelistet, die zur Auswertung und grafischen Aufbereitung benötigt werden.

10.2 Ergebnisse

Analysiert werden die Dienste FASE [2] 2008 (ETAPS Konferenz), ICA 2009 [3] und TACAS 2009 (ETAPS Konferenz). Tabelle 10.1 beinhaltet die Eckdaten der Dienste, die auf dem Springer-eigenem OCS-Cluster gehostet wurden.

Data \ Conferences	FASE'08	ICA'09	TACAS'09	total
Chairs	2	7	2	**11**
Committee (PCC & PCM)	30	51	29	**110**
SubReviewers	61	15	88	**164**
Authors | Submissions	153	134	131	**418**
CoAuthors	216	203	231	**650**
Users	460	359	479	**1298**
Period [d]	108	101	113	
Logfile [MB]	408	451	613	**1472**

Tabelle 10.1: Fallstudie - Eckdaten der analysierten Dienste

Die Fallstudie umfasst die in der Tabelle 10.1 gelisteten Konferenzen mit insgesamt 1298 Benutzern und 418 eingereichten Artikeln. Werden die ETAPS-Konferenzen FASE und TACAS näher betrachtet so haben sich die Einrei-

[1] Comma-Separated Values

[2] 11th International Conference on Fundamental Approaches to Software Engineering

[3] 8th International Conference on Independent Component Analysis and Signal Separation

chungen bei FASE von 2002 bis heute verdreifacht und die Anzahl der Dienstbenutzer versiebenfacht. Tabelle 10.2 zeigt diesen Sachverhalt. Die mit einem Trennstrich markierten Felder weisen Konferenzen auf, die nicht mit dem OCS durchgeführt wurden.

	FASE		**TACAS**	
Jahr	**Einreichungen**	**Benutzer**	**Einreichungen**	**Benutzer**
2000	-	-	140	157
2001	-	-	129	147
2002	56	71	-	-
2003	97	112	154	180
2004	98	**181**	145	**242**
2005	101	184	167	308
2006	137	241	161	305
2007	141	**581**	204	**660**
2008	153	488	174	601
2009	-	-	131	533

Tabelle 10.2: Fallstudie - FASE und TACAS Historie

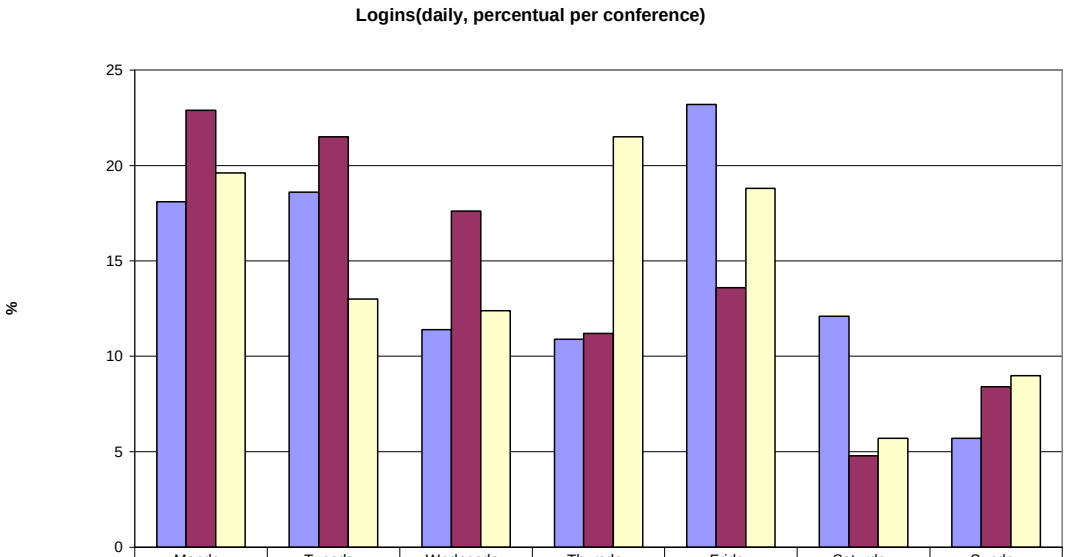
Die Zunahme der Dienstbenutzer ist von mehreren Kriterien abhängig:

- Die zunehmende Größe des Komitees.
- Die Anzahl der Submissions und die daraus folgende Anzahl an Autoren und CoAutoren.
- Die Anzahl der SubReviewer, die hinzukommen, falls die Weiterdelegierung von Gutachten aktiv ist.

Die CoAutoren wurden vorher nur mit deren Namen angegeben und nicht als ein Dienstbenutzer geführt. Heutzutage können diese von den Autoren registriert werden und explizit das Recht zum Einloggen erhalten, um wie der eigentliche Autor zu agieren, wobei die Rechte fein justierbar sind. Dasselbe gilt im Bereich der SubReviewer, die ebenfalls als Benutzer im Dienst aufgenommen wurden, damit der Prozess der Weiterdelegierung transparent für die Konferenzleitung wird. In Tabelle 10.2 ist die Einführung des SubReviewing im Jahre 2003 und die Aktivschaltung des CO-Author-Workflows im Jahre 2006 deutlich an den im darauffolgendem Jahr gestiegenen Benutzerzahlen zu erkennen.

10.2.1 Benutzungsstatistik

Die Benutzung des Dienstes und somit die Arbeitsprofile der Benutzer der jeweiligen Konferenz lassen sich anhand von Benutzersitzungen aufzeigen, die durch die erfolgreichen Anmeldevorgänge (logins) reflektiert werden. TACAS ist mit 6779 Logins die aktivere Konferenz im Vergleich zu FASE mit 4022 und ICA mit 3980 Logins (siehe Tabelle 10.4). Die Abbildung 10.1 stellt die aktiven Benutzersitzungen pro Wochentag prozentual dar.



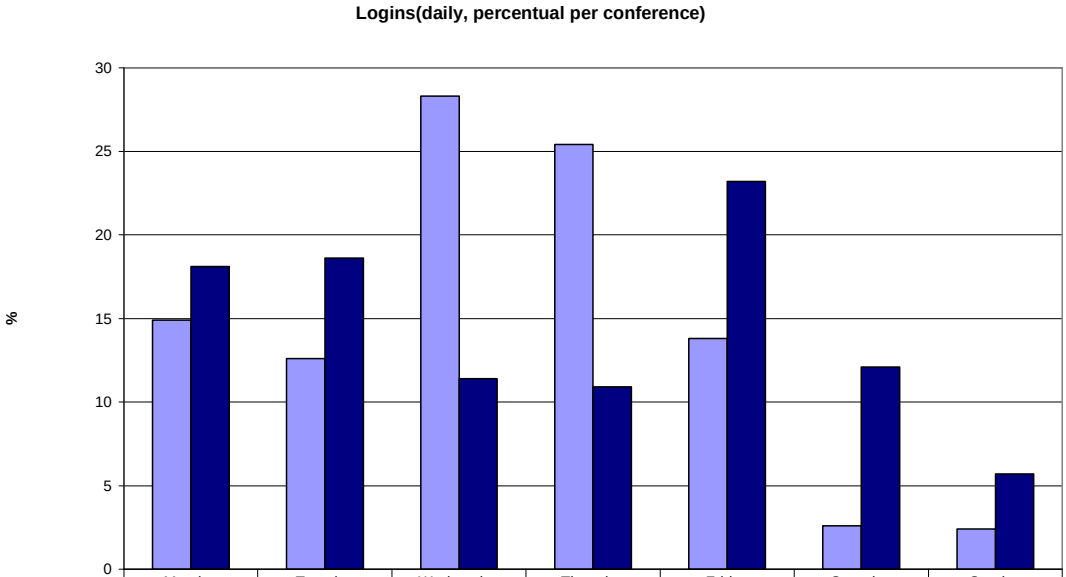
	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
FASE2008	18,1	18,6	11,4	10,9	23,2	12,1	5,7
ICA2009	22,9	21,5	17,6	11,2	13,6	4,8	8,4
TACAS2009	19,6	13	12,4	21,5	18,8	5,7	9

Abbildung 10.1: Fallstudie - Prozentuale Benutzung der Konferenzen: täglich

FASE und ICA stimmen von der Tendenz des Verlaufs der täglichen Logins bis auf den Dienstag überein. TACAS folgt ebenfalls dem Verlauf mit einer Ausnahme am Donnerstag mit entgegengesetztem Trend. Die meisten Benutzersitzungen gab es am Montag mit 2965 und am Freitag mit 2749. Dieses zeigt eine Arbeitsweise, die den Montag für die Nacharbeitung des Wochenendes und zur Vorbereitung der anfangenden Woche nutzt. Der Freitag dient dabei zur Nacharbeitung der Woche und Vorbereitung des Wochenendes. Aus der Erfahrung bekannt ist der Freitag ein beliebter Tag für Deadlines, wie z.B. der letzte Tag für die Einreichung der Artikel. Dieses beeinflusst mitunter die Spitzen im Diagramm. Es ist ein hohes Maß an Aktivität am Wochenende festzustellen (Samstag 1067 und Sonntag 1170), was für Samstag wie Sonntag ein gutes Drittel im Vergleich zum aktivsten Tag dem Montag ist. Die Wochen-

endaktivität bewegt sich somit zwischen 13,2 % und 17,8 %. Diese zeigt die Notwendigkeit eines Systems, das 24x7 Verfügbarkeit aufweist. Der Support kann auf Basis einer solchen Studie und den Erfahrungen geplant werden.

Die Abbildung 10.2 stellt die Konferenzen FASE'02 und FASE'08 gegenüber.



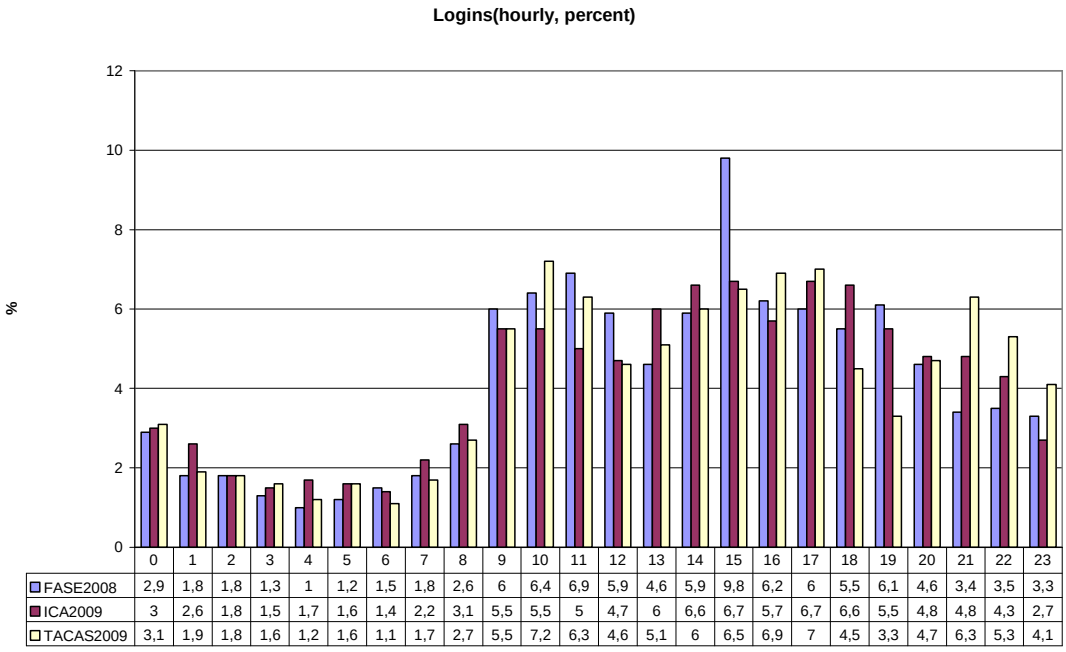
	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
FASE2002	14,9	12,6	28,3	25,4	13,8	2,6	2,4
FASE2008	18,1	18,6	11,4	10,9	23,2	12,1	5,7

Abbildung 10.2: Fallstudie - Vergleich FASE'02 und FASE'08: täglich

Die tägliche Darstellung zeigt für FASE'02 eine Glockenkurve, wobei der Montag mehr Aktivität als der Dienstag aufweist. Die Aktivitäten von FASE'08 zeigt ein unterschiedliches Verhalten auf. Fällt die Hauptaktivität von FASE'02 auf den Mittwoch und Donnerstag, so sind die Haupttage für FASE'08 am Anfang und Ende der Arbeitswoche zu finden. Auffällig ist die um mehr als das Dreifache gestiegende Arbeitsbereitschaft am Wochenende der aktuelleren Konferenz mit 17.8 % zu 5 % im Jahre 2002. Dieses kann auf die Veränderung des Komitees zurückgeführt werden.

Die in Abbildung 10.3 gezeigte stündliche Verteilung der Konferenzen FASE'08, ICA'09 und TACAS'09 weist im Zeitraum von 9 Uhr bis 20 Uhr mit 65,6 % die meisten Aktivitäten auf. Dieser Zeitraum ist mit den um eine Stunde verschobenen Arbeitszeiten in Europa vergleichbar. Die hohe Aktivitäten in den Abend- und Nachtstunden ist auf Forscher aus anderen Zeitzonen, wie z.B. den USA, zurückzuführen. Wird der Zeitraum von 1 Uhr bis 9 Uhr Morgens betrachtet, finden 14,2 % der Gesamtaktivitäten in der europäischen Nacht statt.

Der Zeitraum von 20 Uhr bis 1 Uhr bestimmt 20,2 % der Gesamtaktivitäten.



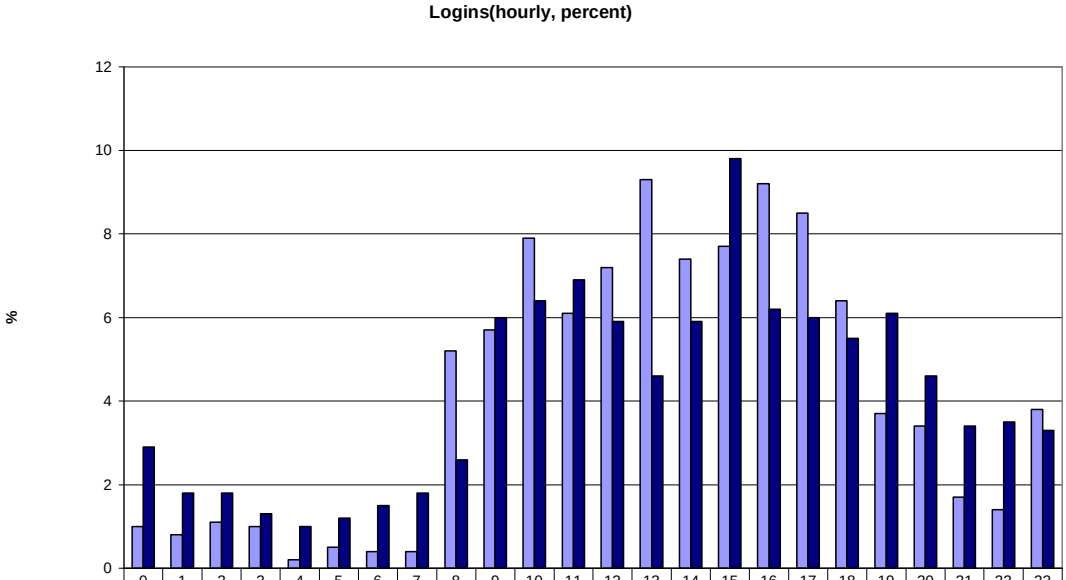
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
FASE2008	2,9	1,8	1,8	1,3	1	1,2	1,5	1,8	2,6	6	6,4	6,9	5,9	4,6	5,9	9,8	6,2	6	5,5	6,1	4,6	3,4	3,5	3,3
ICA2009	3	2,6	1,8	1,5	1,7	1,6	1,4	2,2	3,1	5,5	5,5	5	4,7	6	6,6	6,7	5,7	6,7	6,6	5,5	4,8	4,8	4,3	2,7
TACAS2009	3,1	1,9	1,8	1,6	1,2	1,6	1,1	1,7	2,7	5,5	7,2	6,3	4,6	5,1	6	6,5	6,9	7	4,5	3,3	4,7	6,3	5,3	4,1

Abbildung 10.3: Fallstudie - Prozentuale Benutzung der Konferenzen: stündlich

Werden die stündlichen Auswertungen von FASE'08 mit FASE'02 verglichen, siehe Abbildung 10.4, ist ein ähnlicher Verlauf der Aktivitäten festzustellen. Allerdings sind die Aktivitäten am späten Abend und in der Nacht für FASE'08 ausgeprägter, da das im Jahre 2002 reine europäische Komitee heutzutage 5 Mitglieder aus den USA und eins aus Kanada beinhaltet. Waren es 2002 1745 Logins während der Konferenzlaufzeit sind es 2008 aufgrund der höheren Benutzerzahlen 4022 gewesen.

10.2.2 Feature-Statistik

Im Folgenden wird auf die Verwendung der Dienst-Features eingegangen. Anhand dieser Statistik kann erkannt werden, ob und wie oft ein Feature zum Einsatz kam. Die Tabelle 10.3 beinhaltet die Auswertung der Features der betrachteten Konferenzen. Dabei steht PCC für PC Chair, PCM für PC Member, SR für SubReviewer und A for Author. Es werden die gesamten Zugriffe auf ein Feature, die prozentuale Anteil und die durchschnittliche Benutzung eines Features von einem Benutzer aufgelistet. Die Nullwerte in der Tabelle besagen, dass ICA sowohl das Bidding sowie das Forum Feature nicht benutzt hat. ICA



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
FASE2002	1	0,8	1,1	1	0,2	0,5	0,4	0,4	5,2	5,7	7,9	6,1	7,2	9,3	7,4	7,7	9,2	8,5	6,4	3,7	3,4	1,7	1,4	3,8
FASE2008	2,9	1,8	1,8	1,3	1	1,2	1,5	1,8	2,6	6	6,4	6,9	5,9	4,6	5,9	9,8	6,2	6	5,5	6,1	4,6	3,4	3,5	3,3

Abbildung 10.4: Fallstudie - Vergleich FASE'02 und FASE'08: stündlich

hat ein reines reales PC-Treffen (PC Meeting) unter Verwendung des **Evaluation**-Features durchgeführt.

Nachstehend werden einige der aufgelisteten Feature aufgrund ihrer Ergebnisse diskutiert.

Das **Article**-Feature ist mit im Durchschnitt 34,8 % aller Zugriffe auf die drei Konferenzen das meist genutzte Feature der betrachteten Entscheidungsdienste. Es wird gefolgt vom Forum mit 30 % und der Tasklist mit 16,5 %. Im Vergleich zu FASE'02 ist die durchschnittliche Nutzung des Article-Features durch einen Benutzer (apu) von 78 auf 15 Zugriffe gesunken. Dieses liegt mit unter an der Einführung des Bidding-Features.

Das **Bidding**-Feature erleichtert die Verteilung der Begutachtungsaufgaben und reduziert die Last der Konferenzleiter und des Article Feature, da der im Article-Feature eingeordnete Delegierungsworkflow nur noch teilweise zum Einsatz kommt. Das Feature wurde von FASE'08 und TACAS'09 erfolgreich eingesetzt.

Das **Evaluation**-Feature wurde mit 4,1 % am häufigsten von TACAS eingesetzt, was als Konsequenz aus der regen Diskussionsbereitschaft von 28,9 % zu sehen ist. Die Zwischen- und Endergebnisse der Diskussion werden von der Konferenzleitung in die Evaluationmatrix übertragen. Dieses geschieht durch

Feature	Roles	FASE2008				ICA2009				TACAS2009			
		users	total	%	apu	users	total	%	apu	users	total	%	apu
Articles	PCC,PCM,A	183	2823	24,2	15	185	4352	51,2	24	160	6002	29	38
Bidding	PCC, PCM	30	243	2,1	8	51	0	0	0	29	342	1,7	12
Conflicts	PCC	2	20	0,2	10	7	43	0,5	6	2	69	0,3	35
Delegation	PCC, PCM	30	172	1,5	6	51	217	2,6	4	29	430	2,1	15
Email	PCC	2	31	0,3	16	7	110	1,3	16	2	137	0,7	69
Evaluation	PCC,PCM	30	358	3,1	12	51	142	1,7	3	29	852	4,1	29
Forum	PCC,PCM	30	3628	31,1	121	51	0	0	0	29	5980	28,9	206
Preferences	PCC, PCM	30	243	2,1	8	51	269	3,2	5	29	288	1,4	10
Profile	PCC,PCM,A	183	243	2,1	1	185	367	4,4	2	160	381	1,8	2
Reports	PCC,PCM	30	862	7,4	29	51	412	4,8	8	29	1730	8,4	60
Roles	PCC	2	33	0,3	17	7	55	0,7	8	2	62	0,3	31
Setup	PCC	2	96	0,8	48	7	268	3,2	38	2	135	0,7	68
Staff	PCC, PCM	30	186	1,6	6	51	249	2,9	5	29	246	1,2	8
Tasklist	PCM, SR	117	2242	19,2	19	59	1314	15,5	22	169	3072	14,9	18
Users	PCC	2	475	4,1	238	7	699	8,2	100	2	942	4,6	471
total			**11655**	**100**	**554**		**8497**	**100**	**241**		**20668**	**100**	**1072**
apu indicates the average usage per user										**u** = users			

Tabelle 10.3: Fallstudie - Statistik der Feature

das Setzen des Artikels in den entsprechenden Status. In Abhängigkeit der in 2002 berechneten Benutzung des Evaluation-Features und die hohe Akzeptanz des Features wurde die Forumkomponente als ein Hauptfeature in das Entscheidungssystem integriert. Diese Design-Entscheidung hat zur Verbesserung der Ergonomie beigetragen.

Der hohen apu-Werte des **Forum**-Features kennzeichnet dieses als besonders wichtig für die Konferenzen FASE und TACAS. Die Gesamtzugriffszahlen von FASE mit 3628 und TACAS mit 5980 weisen auf eine rege Diskussion zur Festlegung der angenommenen bzw. abgelehnten Artikel hin. Mit einem apu-Wert von 121 für FASE und 206 für TACAS gehört das Forum-Feature mit zu den meist genutzten Features.

Das **User**-Feature ist für den PC Chair ein häufig verwendetes Feature. Die Studie von 2002 zeigte für FASE'02 einen apu-Wert von 635 und für die ebenfalls analysierte Konferenz ESOP'02 von 1072. Dies war der Anlass die Nachfrage für ein neues Passwort zu automatisieren. Die in späterer Kombination mit dem Ticketsystem, das direkte Einstiege in das System ermöglicht, ließ die Benutzung des User-Features von 13,3 % (in 2002) auf heutige 4,1 % sinken. Für die PC Chairs stellt dies eine erhebliche Arbeitserleichterung dar, weil diese als alleinigen Benutzer des Features angesehen werden können. Die eventuelle Beinflussung durch die Administratorrolle ist aufgrund des sehr geringen Aktivitäten zu vernachlässigen. Die Einführung des Ticketsystem hat

den Workflow im ergonomischen Sinne aufgewertet.

Die TACAS Konferenz erweist sich bei der Verwendung des **Setup**-Features (apu 68) und **Role**-Features (apu 31)als die aktivere Konferenz in Bezug auf die Anpassung des Dienstes an die Konferenz. Die Auswertung der beiden Features zeigt bei allen drei Konferenzen die Ausnutzung des flexiblen, zur Laufzeit konfigurierbaren Ansatzes des Entscheidungssystems.

10.2.3 Konferenzstatistik

Nach der Veranschaulichung der Benutzungs- und Feature-Statistik wird in diesem Kapitel kurz auf die globalen Ergebnisse der Konferenzen eingegangen. In Tabelle 10.4 sind die wesentlichen Daten zur Diskussion zusammengefasst.

Data	FASE'08	ICA'09	TACAS'09
Logins	4022	3980	6779
Automatic Emails	3029	615	3302
Uploads	281	333	448
Downloads	913	1146	1761
SubReports	95	28	119
Reports	364	346	407
Final Reports	118	134	131
News (post)	387	0	448
News (read)	3241	0	5532
News total use	3628	0	5980
Pwd. Changes	6	20	11
Users	494	366	579
PWD Index[%]	1,2	5,5	1,9

Tabelle 10.4: Fallstudie - Globale Daten der Konferenzen

TACAS war eine sehr umfangreiche und aktive Konferenz, die mit 6779 Logins vor FASE mit 4022 und ICA mit 3980 liegt. An der sehr hohen Benutzung des Forum-Features zeigt sich eine rege Diskussion bei der Festlegung über Annahme oder Ablehnung der begutachteten Artikel. Mit 448 Postings und einer Gesamtnutzung des Features von 5980 Zugriffen liegt TACAS vor FASE (387, 3628). Auffällig ist die hohe Anzahl von 1761 Artikel die heruntergeladenen wurden. Dies kann mit der Einreichung von neueren Versionen eines Artikels durch die Autoren zusammenhängen, die erneut von den Gutachter heruntergeladen worden sind. Mit 579 Benutzern ist TACAS ebenfalls die größte Konferenz vor FASE (494) und ICA (366). Die hohe Anzahl von 3302 versendeten Emails umfassen neben den Benachrichtigungen an den PC Chair über

den bevorstehenden Ablauf einer Konferenzphase, hauptsächlich die Warnungen über verstrichene Begutachtungstermine seitens der Gutachter. 119 SubReports bestätigen die Akzeptanz des Workflows zur Weiterdelegierung der Gutachten.

Auffällig sind 407 Reports bei TACAS mit 131 Artikeln im Vergleich zu ICA mit 346 Reports für 153 Artikel. Dieses bedeutet eine höhere Anzahl von Gutachtern pro Artikel auf Seiten von TACAS. Bei ICA wurde die Weiterdelegierung mit 28 an der Zahl nur gering eingesetzt. Die geringe Anzahl von 615 Benachrichtigungsemails (Automatic Emails) weist auf eine organisierte Begutachtungsphase hin, in der sich die Gutachter an die Termine für die Abgabe der Gutachten gehalten haben. Die Diskussion über die Artikel wurde gänzlich während des Komiteetreffens (PC meeting) durchgeführt.

FASE verfügt mit 153 Artikel über die meisten Einreichungen, vor ICA mit 134 und TACAS mit 131 Artikeln. Die hohe Anzahl der automatisch veschickten Emails zeigt wie bei TACAS eine vielfache Überschreitung von Begutachtungsterminen auf. An den hohen Werten bzgl. des Forum-Features ist auch hier eine rege Diskussion festzustellen.

10.3 Fazit

Wie zu Beginn dieses Kapitels erwähnt, lassen sich aus der Durchführung einer solchen Studie verschiedene Erkenntnisse gewinnen. Dieses ist unter anderem die Auslastung eines Konferenzdienstes durch die Anzahl der Dienstbenutzer und die daraus notwendige Dimensionierung des Server-Clusters. Die Anzahl der Dienstbenutzer liegt im Durchschnitt bei 500 Benutzern pro Konferenz, wie in Tabelle 10.2 zu sehen ist, und durch weitere ähnlich dimensionierte Konferenzen bestätigt wird. Die Erhöhung der Anzahl der Benutzer hängt mit steigenden Einreichungszahlen und die direkte Integration von Co-Autoren (im Jahr 2006) und SubReviewern (im Jahr 2003) in den Begutachtungsprozess zusammen, was an den gestiegenden Benutzerzahlen des darauffolgendem Jahr zu erkennen ist. Zum anderen wird durch die Auswertung der täglichen und wöchentlichen Benutzung der Konferenzendienste aufgezeigt, wie notwendig eine durchgehende *24x7* Verfügbarkeit der Dienste ist. Das wird einerseits durch die Aktivitätsverteilung über den Tag mit 65,6 % von 9 Uhr bis 20 Uhr, mit 20,2 % von 20 Uhr bis 1 Uhr und mit 14,2 % von 1 Uhr bis 9 Uhr und andererseits durch eine Wochenendaktivität von 13,2 % bis 17,8 % der Gesamtwochenaktivität untermauert.

Eine weitere Erkenntnis ist die Akzeptanz der Features und eventuell ein-

hergehenden Änderungen von Design-Entscheidungen. Eine wichtige Stellung nimmt die Beobachtung ein, wie der Dienst auf unterschiedliche Weise eingesetzt wird. Features können anhand der Anzahl von Benutzungen festgemacht und im Hinblick auf die Wichtigkeit klassifiziert werden. So wird die Einführung des Workflows zur Weiterdelegierung der Gutachten an SubReviewer mit einer Gesamtzahl von 242 SubReports bestätigt und von den PC Membern sehr geschätzt.

Ein weiteres zentrales Feature ist das Forum-Feature, das im Jahre 2000 von mir auf Basis eines News-Servers [4] und dem Protokoll NNTP [5] als Graphmodell umgesetzt und in den OCS integriert wurde. Die Integration einer Diskussionkomponente in ein Begutachtungssystem revolutionierte den Prozess der Entscheidungsfindung über Annahme bzw. Ablehnung der eingereichten Artikel und nahm lange Zeit eine Vorreiterstellung ein. Stellten die PC-Meetings zur Feststellung der zu akzeptierenden Artikel einen sehr hohen Kostenfaktor und Zeitaufwand dar, konnte die Entscheidungsfindung fortan integriert im OCS stattfinden. Zugriffe auf die einzelnen Foren, jeder Artikel hat sein Eigenes, konnten gezielt freigeschaltet werden, wobei der Dienst bereits automatisch die jeweiligen Rechte setzte. Die Umsetzung einer integrierten Komponente zur *virtuellen* Durchführung von PC-Meetings senkte den Kosten- und Zeitaufwand für das Komitee und führte zu einer beschleunigten und zielgerichteten Entscheidungsfindung. Das Forum-Feature erfreute sich sehr schnell größter Beliebtheit [MK02] und wird weiterhin für das *virtuelle* PC-Meeting eingesetzt, siehe Tabelle 10.3. Kommunities, die ihre PC-Meetings nach wie vor *physikalisch* durchführen, verwenden das Forum-Feature um Vorentscheidungen zu treffen, die Grundlage eines verschlankten *physikalischen* PC-Meetings sind. Der große Nutzen und Erfolg hatte dazu beigetragen, dass die Idee einer integrierten Diskussionskomponente durch die Entwickler anderer Begutachtungssystemen aufgegriffen wurde und sich zum Standard-Feature entwickelte. Die aufgezeigte Umstrukturierung der Feature-Hierarchie im OCS in Bezug auf das Forum-Feature hat einen weiteren Schritt in Richtung einer guten Ergonomie des OCS bedeutet. Das Forum-Feature wurde zu einem Haupt-Feature umgewandelt, so dass es in Kombination mit dem eingeführten Ticketsystem zu den meist genutzten Feature geworden ist. Dieses ist durch einem apu [6]-Wert von 121 für FASE und 206 für TACAS in Tabelle 10.3 dokumentiert. Der apu-Wert ist von 39 für FASE'02 auf 121 für FASE'08 gestiegen, was eine Erhöhung der Benutzung des Forum-Features unterstreicht. Ein anderes Beispiel für die Akzeptanz einer solchen Studie ist die Benutzung des User-Features von 13,3

[4] INN: InterNetNews

[5] Network News Transfer Protocol

[6] Durchschnittliche Benutzung eines Features pro Benutzer.

% in FASE'02, die durch eine daraufhin stattgefundene Umsetzung einer automatischen Passwortvergabe auf 4,1 % in FASE'08 gesunken ist. Dieses stellte sich über die Jahre als Arbeitserleichtung für die PC Chairs heraus.

Durch das Wissen, wie sich ein System in anderen Kontexten verhält bzw. eingesetzt wird, kann es für ein weites Spektrum gezielt entwickelt und an verschiedene Geschäftsprozesse angepasst werden.

Teil V

Zusammenfassung und Ausblick

Kapitel 11

Zusammenfassung und Ausblick

11.1 Zusammenfassung

In dieser Arbeit wurde insbesondere eine graphbasierte und Feature-orientierte Vorgehensweise für die Software-Entwicklung im Rahmen einer konkreten praktischen Umsetzung vorgestellt, die unabhängig von der für das zu konzipierende und realisierende Projekt verwendeten Technologie ist. Der *Online Conference Service* ist 1999 auf Basis des beschriebenen Konzeptes Service-orientiert entwickelt worden und kam erstmalig 2000 in direkter Kooperation mit Springer Heidelberg für eine internationale Konferenz zum Einsatz (siehe Tabelle 9.1). Im Laufe der Jahre vergrößerte sich der Umfang an Dienstfunktionalitäten ständig, bis der OCS 2003 eines der flexibelsten Begutachtungssysteme war, das sich an verschiedene Konferenzanforderungen anpassen konnte. Das vorgestellte Konzept basiert auf dem modellgetriebenen Ansatz des ABC und wurde bereits Ende der neunziger Jahre entwickelt. Das Konzept stellte zu dieser Zeit einen neuartigen Ansatz für die Realisierung von komplexen Web-Anwendungen im Bereich von Begutachtungssystemen dar. Die realisierte Service-orientierte Vorgehensweise ermöglicht Geschäftsprozesse von Unternehmen hierarchisch als Features zu strukturieren und in Form von Graphen abzubilden, die direkt den Kontrollfluss der Anwendung repräsentieren. Auf diese Weise entstehen übersichtliche, in Haupt-Features und Unter-Features gekapselte Prozessketten, die ausführbar sind. Durch die vollautomatische Code-Generierung des ablauffähigen Kontrollflusses ist gewährleistet, dass die Modelle der Geschäftprozesse mit der Code-Repräsentation jederzeit konform sind, was die Agilität in Bezug auf die Anpassung bestehender Graphmodelle und respektive der Ausführungsebene hervorhebt. Die modellgetriebene und agile Vorgehensweise unterstützt den gesamten Entwicklungsprozess und Le-

benszyklus einer Applikation, wie in Abbildung 11.1 (aus [MS06]) dargestellt.

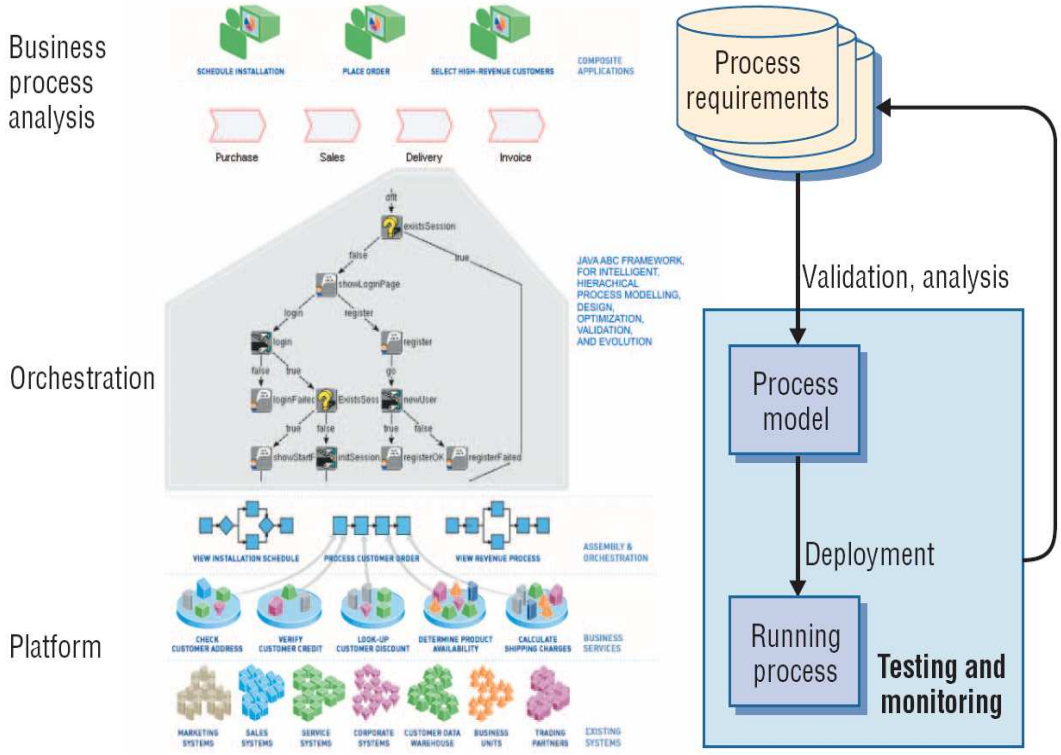


Abbildung 11.1: Agiles Prozessorientiertes Design

Kundengespräche bzgl. der Geschäftsabläufe können anhand der Graphebene erfolgen und auftretende Änderungen der Prozesse direkt auf der Ebene eingepflegt werden. Sind Modifikationen auf Basis von existierenden Dienstkomponenten durchzuführen, so ist der neue Geschäftsablauf nach der Generierung der Code-Repräsentation direkt ausführbar und steht für die Kundenpräsentation und die Installation direkt zur Verfügung. Sind hingegen neue Dienste für die Umsetzung des geänderten Geschäftsprozesses zu implementieren, dient die graphbasierte Modellierung des neuen Workflows als Grundlage zur Kommunikation und Diskussion mit dem Kunden und später mit dem SIB-Entwickler, was mit einer deutlichen Vereinfachung der Anwendungsentwicklung durch die Visualisierung der Geschäftsprozesse und der direkten Code-Generierung einhergeht. Die vom Kunden (Applikationsexperten) orchestrierten Geschäftsprozesse zeigen dem SIB-Entwickler (Programierexperte) einerseits die zu realisierenden Dienste (SIBs) auf, andererseits sind die Dienste bereits zu einem Kontrollfluss orchestriert, was dem Entwickler die Zusammenhänge der Services darstellt. Kunden werden direkt in den Entwicklungsprozess der Applikationen

eingebunden und in die Lage versetzt, Geschäftsprozesse ohne Programmierkenntnisse zu modellieren und bedingt durch die Änderung von Geschäftsabläufen bzw. Firmenstrukturen, existierende Graphmodelle selbständig zu modifizieren, um die Prozesse der Applikation an die neuen Gegebenheiten einfach und schnell zu adaptieren. Aus mehrjähriger Erfahrung ist es von großer Bedeutung Kunden mit in die Appliaktionsentwicklung einzubeziehen, um die selbständige Erstellung und vor allem Anpassung von Geschäftsmodellen zu ermöglichen und das bereits direkt zur Modellierungsphase der Applikation. Modellierte Geschäftsprozesse werden für den Auftraggeber verständlicher, was die Kommunikation zwischen der Geschäftswelt und den IT-Experten deutlich vereinfacht.

Andere Prozessorientierte Ansätze, wie das 1993 von Prof. August-Wilhelm Scheer eingeführte ARIS [1]-Konzept [Sei02] basierend auf der verbreiteten Modellierungssprache *eEPK* [2], setzen UML-Werkzeuge für die Code-Generierung ein. Im Vergleich zum eigenen Ansatz war für die ARIS-Modelle auf der Ebene der Geschäftsprozesse keine direkte vollständige Transformation in eine Code-Repräsentation als ausführbaren Kontrollfluss vorgesehen. Heutzutage verfügt der ARIS UML Designer, der das ARIS-Konzept an UML anbindet, zwar über ein Plugin für die Verwendung des Code-Generator Frameworks openArchitectureWare [oT09] [And06], allerdings ist ein solcher Ansatz von zu technischer Natur, um Kunden bzw. Geschäftspartner ohne Programmierkenntnisse in den Entwicklungsprozess zu integrieren. Derartige UML-basierte Ansätzen, zu denen auch Werkzeuge wie *Together* oder *Rational Software Architect* gehören, verwenden Code-Generator-Frameworks wie openArchitectureWare oder AndroMDA [Tea09a](angelehnt an OMG's Model Driven Architecure) zur Generierung von Sourcecode. Der überwiegende Teil des generierten Codes ist unvollständig (nur Code-Rahmen) und muss von Entwicklern implementiert werden, was mit dem Problem des RTE [3] einhergeht. Änderungen die direkt am generierten Sourcecode durchgeführt werden, sind in das entsprechende Modell einzuarbeiten, um eine Konsistenz zwischen Modell und Sourcecode zu gewährleisten. Das in Abbildung 11.2 (aus [SJWM09]) dargestellt Bild *b)* illustriert dieses Szenario. In komplexen Anwendungen mit verschiedenen Modellen, wie es bei UML der Fall ist, stellt sich ein permanenter Abgleich als ein sehr großer Aufwand und oftmals schwer zu realisierender Prozess dar.

Der Fokus der eigenen Vorgehenweise der Softwareentwicklung liegt in der Einfachheit und der Verständlichkeit komplexe Anwendungen auf Geschäftsprozessebene durch Graphmodelle abzubilden, schnell an neue Geschäftsabläufe

[1] Architektur Integrierter Informationssysteme
[2] *erweiterte Ereignisgesteuerte Prozesskette* von Prof. August-Wilhelm Scheer
[3] Roundtrip Engineering

zu adaptieren (Agilität) und durch direkte vollautomatische Code-Generierung (ohne manuelles Eingreifen in den generierten Code) eine ausführbare Applikation zu erstellen bzw. Änderungen in eine bereits existierende Anwendung schnell und sicher (Verfikation der Graphmodelle) einzupflegen. Die Ansätze OTA [4] [SJWM09, MS09, SN07] und XMDD [5] [SJWM09, MS08] stehen für die Umsetzung derartiger Anforderungen, zu deren Erforschung Projekte wie der OCS in Form einer Fallstudie maßgeblich beigetragen haben. OTA zeigt den gesamten Entwicklungsprozess vereinfacht als ein Modell auf, das für Applikationsexperten und Geschäftsleute (Nicht-IT-Experten) gleichermaßen verständlich und transparent ist. XMDD ist eine extreme Ausrichtung des MDD [6], die alle Geschäftsprozesse einer Applikation auf der Modellierungsebene beschreibt. Des Weiteren gewährleistet der Ansatz die Konsistenz zwischen dem Modell und dem generierten Sourcecode, was ein wesentlicher Aspekt für eine agile, Modellgetriebene Softwareentwicklung ist (siehe Abbildung 11.2 Bild *c)*).

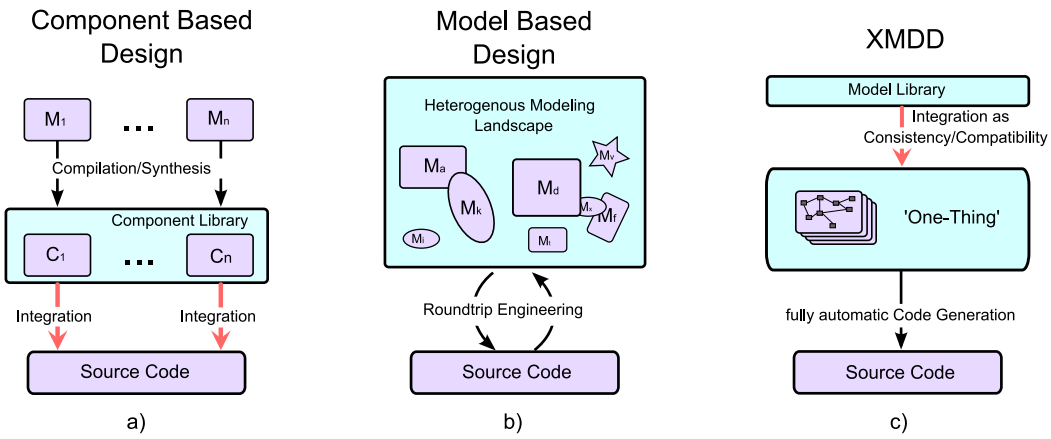


Abbildung 11.2: Component-, Model-Based Design, and XMDD

Änderungen bzgl. der Geschäftsprozesse werden ausschließlich auf Modellierungsebene durchgeführt, wodurch das RTE hinfällig wird. Der generierte Sourcecode stimmt zu jeder Zeit mit dem Modell überein und ist konsistent. Diese Vorgehensweise in Kombination mit dem Feature-orientierten Ansatz machten den OCS zu einem sehr agilen System in Bezug auf die Adaption des Geschäftsprozesses. Das Bild *a)* in Abbildung 11.2 zeigt die Vorgehenweise bei dem *Component Based Design*. Es existieren Modelle aus denen jeweils eine

[4] The One-Thing-Approach
[5] eXtreme Model-Driven Design
[6] Model-Driven Design

Komponente generiert wird, wobei die Modelle nicht miteinander verbunden sind. Die Umsetzung der Applikation findet auf Basis der Bibliotheken von Komponenten statt und nicht auf der Modellierungsebene, was dem heutigen Konzept der Softwareentwicklung entsprechen würde.

Durch den vorgestellten Zugriffskontrollmechanismus [KM06], der sich nahtlos in das Feature-orientierte Konzept [KM05] einfügt, können Zugriffe auf die Dienstkomponenten auf einfache Weise gesteuert und die Benutzersichten personalisiert werden. Dieses realisierte Rollen/Rechte Management, das für dynamische Rollen und Rechteabhängigkeiten realisiert wurde, zeichnet sich durch seine hohe Flexibilität aus. Die Rechtezuordnungen in Begutachtungssystemen, insbesondere für Journals, erfordern eine höhere Dynamik als in anderen Standard RBAC-Anwendungen [FK92]. Auf eine hierarchische Strukturierung der Rollen, wie sie in [SCFY96] vorgestellt wird, ist aufgrund der sich dynamisch ändernden Zugriffsrechte in Begutachtungssystemen bei der Realisierung des Zugriffskontrollmechanismus von der Hierarchie abgesehen worden. Eine Hierarchisierung hätte zufolge gehabt, dass für jede zusätzliche Rolle bzw. für jede spezielle Anpassung einer Rolle für eine Einzelperson, der Umfang der Hierarchie zunimmt, was letztendlich zu einer Komplexität der Verschachtelungen in der Hierarchiestruktur geführt hätte, die schwer zu konfigurieren und warten wäre. Das Rollen/Rechte Management wurde XACML [7] [Mos05] und WS-Policy [BNB+06, WCL+05] gegenübergestellt [KMW08] und aufgezeigt, wie zentral eine Verifikation von Policies (constraints) für die Gültigkeit von Rechtesysteme ist.

Die Verifikation eines Graphmodells wird mittels eines Modelcheckers durchgeführt, der ein Modul des ABC ist. Der Kontrollfluss (Geschäftslogik) des OCS ist als gerichtete Graphen modelliert, die auf zentrale Eigenschaften hin überprüft werden können [KM05, KM06, KMW08]. Hierfür werden atomare Eigenschaften (Propositionen) an die Knoten der Graphen annotiert. Die Verifikation der Systembeschreibung (Graphmodell) wird gegen eine Spezifikation durchgeführt. Diese Spezifikation wird als temporal logische Formel in CTL (Computation Tree Logic) geschrieben (siehe Kapitel 2.2.3), die auf den annotierten Propositionen aufbaut. Der Modelchecker überprüft diese Grapheigenschaften und zeigt vorhandene Inkonsistenzen auf.

Die Umsetzung des OCS und seiner Dienstfamilie [KM06] (siehe Kapitel 7) unter Einhaltung der beschriebenen Vorgehensweisen verfolgte bereits Ende der Neunzigerjahre eine Service-orientierte Softwareentwicklung und zeigte eine Lösung für die Umsetzung komplexer Anwendungen für ausführbare Geschäftsprozesse auf, die erst später, etwa durch die im Jahre 2002 eingeführ-

[7] eXtensible Access Control Markup Language

te Sprache für Geschäftsprozesse WS-BPEL, in den Fokus intensiver Forschung und einer breiten öffentlichen Beachtung gelangten. IBM, BEA Systems und Microsoft realisierten die XML-basierte Sprache WS-BPEL [GKA+07], die zur reinen Orchestrierung von *Web Services* dient. BPEL basiert auf den Sprachen XLANG [Mic09c] von Microsoft und WSFL [8] [Wes09] von IBM. Im Gegensatz zum ABC mit seiner für Web-Dienste konzipierten EWIS-Umgebung, die zur Laufzeit mit einer eigenen, leichtgewichtigten Prozess-Engine den Workflow steuert und Benutzerinteraktionen zulässt, sah WS-BPEL keine direkten Interaktionen mit Menschen vor. Erst die Erweiterung WS-BPEL4People realisiert die Interaktion des Menschen im Jahre 2005 unter Verwendung von WS-HumanTask [FAAD07], die seit 2007 zur Standardisierung bei OASIS [9] vorliegt. Damit Sub-Prozesse von BPEL unterstützt werden können ist die Erweiterung BPEL-SPE [KKL+05] 2005 eingeführt worden. Hierarchien von Prozessketten waren bereits seit Entwicklungsbeginn des OCS kontinuierlich durch Einhaltung des Feature-orientierten Ansatzes existent. Im industriellen Bereich hat sich BPEL zu einem Standard zur Beschreibung von ausführbaren Geschäftsprozessen etabliert. Als Schnittstelle und zur Beschreibung der Funktionalitäten von *Web Services* wird WSDL [10] verwendet. Zum Nachrichtenaustausch kommt SOAP [11] [Erl04] zum Einsatz. Im Gegensatz zum ABC, das ein einheitliches Entwicklungskonzept aufweist, das die Geschäftswelt mit der IT-Welt verbindet, verfügt BPEL über kein eigenes Modellierungswerkzeug und keine eigene Laufzeitumgebung (process engine). Die im Jahre 2002 von der BPMI [12] veröffentlichte grafische Notation BPMN [13] [WM08], ist seit 2006 offizieller OMG-Standard für die Modellierung von Geschäftsprozessen. Die erstellten Modelle lassen sich in Ausführungssprachen wie der BPML [14] oder BPEL4WS [15] exportieren, wobei sich BPML nicht gegen BPEL durchsetzen konnte. XPDL [16] ist eine weitere Prozessbeschreibungssprache, die von der WfMC [17] entwickelt wird. Rückblickend lässt sich in Bezug auf die Entwicklung und industrielle Standardisierung von BPEL und der Standardisierung von BPMN feststellen, dass der Ansatz bestehend aus graphbasierter Feature-orientierter Modellierung von hierarchischen ausführbaren Prozessketten auch nach vielen Jahren nichts an seiner Aktualität eingebüßt hat. Das schnelle An-

[8] Web Services Flow Language
[9] Organization for the Advancement of Structured Information Standards
[10] Web Service Description Language
[11] Simple Object Access Protocol
[12] Business Process Management Initiative
[13] Business Process Modeling Notation
[14] Business Process Modeling Language
[15] Business Process Execution Language for Web Services
[16] XML Process Definition Language
[17] Workflow Management Coalition

passen von laufenden Diensten durch den Feature-orientierten Ansatz hat sich in vielen Fällen bewährt, so konnten bestehende Workflows auf einfache Weise modifiziert sowie neue Geschäftsprozesse eingepflegt werden. Kommunities stellten oftmals zu Beginn der Konferenz Anforderungen für neue Features, die umgesetzt und in den laufenden Dienst eingebaut wurden. Hierzu zählt z.B. das Feature für die Anonymisierung der PC Member, das in dieser Arbeit vorgestellt wurde. Diese Anforderung wurde erstmalig von dem Kommittee der Konferenz ATPN an den OCS gestellt. PC Members sollten im Dienst nicht erkennen können, von wem etwas begutachtet oder im Forum diskutiert wurde. Der eigentliche Begutachtungsprozess zwischen den PC Members verläuft anonymisiert. Die Änderungen betrafen mehrere Features des Entscheidungssystems, die rechtzeitig zur Begutachtungsphase eingespielt werden konnten.

Die Vorstellung der Patterns aus *Identify the Champion* von Oscar Nierstrasz [Nie00], die ein Vorgehensmuster für eine effiziente Entscheidungsfindung beschreiben und der anschliessende Vergleich mit den Vorgehensweisen des Entscheidungssystems OCS wiesen Übereinstimmungen auf. Durch die Flexibilität der Features der eigenen Entscheidungssyteme sind diese Patterns ebenso umsetzbar, wie die verschiedenen Ansprüche an die Systemfunktionalitäten durch die Kommunities. Bereits im Jahr 2003 war mit dem OCS ein Begutachtungssystem im Einsatz, das zu diesem Zeitpunkt bereits den Service-orientierten Ansatz verfolgte, der erst später vom Mainstream aufgegriffen wurde. Insbesondere die Agilität und der Funktionsumfang des Systems sind hervorzuheben.

Die Gegenüberstellung der heutigen bekannten Begutachtungssysteme hat gezeigt, dass mittlerweile die zentralen Features auch in andere Systeme Einzug gehalten haben, wobei es signifikante Unterschiede in Bezug auf Umsetzung und Ausprägung gibt. OCS hat, bedingt durch den parallel entstandenen OJS, bereits frühzeitig (zur Entstehungszeit) die meisten der heutigen gängigen Features zur Verfügung gestellt und nahm eine Vorreiterstellung ein, was die Entwicklung einiger Systeme beeinflusst hat. Durch das Feature-orientierte graphische Konzept konnten die jeweils entwickelten Features im OCS sowie im OJS eingesetzt werden. Dieses führte zu einer größeren Anzahl von wiederverwendbaren Komponenten, die ihren Einsatz ebenso in den später realisierten Entscheidungssystemen *LNCS* und *Transactions* fanden. Mit dem Dienst *LNCS* wird über die Zulassung eines Projektes, wie z.B. einer Konferenz oder Journals, zur Publikation entschieden. Der Dienst *Transactions* spiegelt den Entscheidungsworkflow über die Reihe Transactions bei Springer wider. Der OCS wurde Kunden-orientiert entwickelt. Dieses hatte zur Folge, dass durch die damaligen Anforderung der Kunden jedes Detail selbst einstellen zu können, die Komplexität der Konfiguration zu nahm. In den letzten Jahren hat ein Umdenken bei den Kunden stattgefunden, dass Konfigurationen bzw. Einstel-

lungen bzgl. des Begutachtungssystems und dessen Workflows nur noch minimalistischer Natur sein sollen. Die Adaption des Begutachtungssystems auf die spezifischen Anforderungen der verschiedenen Konferenzen ist durch die Auswahl vordefinierter Standardkonfigurationen, inklusive der Konfigurationen der Vorgängerkonferenzen, erwünscht. Beim OCS wurde im Gegensatz zu den anderen Begutachtungssystemen ein Modellgetriebener, Service-orientierter Ansatz [SMN+07, MS04a, KM05, KM06] verfolgt, der neuartig war und heute sehr verbreitet und aktuell ist. Konzepte und Technologien waren im Vergleich zu heute rar oder mitunter noch nicht existent. Dieses führte zu einem robusten und komplexen System, das beim Springer Verlag Heidelberg für die publizierenden Konferenzen in der Fachreihe LNCS seit 2000 als Begutachtungssystem in Kooperation mit dem *Lehrstuhl für Programmiersysteme* angeboten [Hei09b] wird. Im Jahr 2007 wurde die Verwaltung des OCS von Springer übernommen und seitdem von Springer SBM [18] in Dordrecht, Niederlande, gehostet.

Steckten die Technologien und Architekturen für die Realisierung von komplexen Web-Applikationen Ende der Neunzigerjahre noch in den Kinderschuhen oder wurden diese teilweise erst später konzipiert, so hat man heute eine nahezu freie Auswahl im kommerziellen sowie im Open Source Bereich. Das bedeutete für den OCS, dass Komponenten damals von Grund auf selbst implementiert werden mussten. Z.B. wurde die Datenbankschicht direkt auf Basis der Datenbankschnittstelle JDBC [19] [mic09j] realisiert, wobei die Datenbankinteraktionen in den Geschäftsobjekten und Administratoren gekapselt wurden. Diese dienten zur Implementierung der einzelnen Geschäftsprozesselemente (SIB), aus denen die Geschäftslogik der Anwendung graphbasiert modelliert wird. Für die GUI kam WML [WML09] in Verbindung mit Velocity [Apa09] zum Einsatz. Die Personalisierung ist mittels des in dieser Arbeit beschriebenen Zugriffskonzepts durchgeführt und die Geschäftslogik mittels des ABC und EWIS als ausführbarer Kontrollflussgraph modelliert und zu Servlets generiert worden.

Mit dem OCS ist über die Jahre eine Produktlinie [KM06] entstanden, die auf Basis von innovativen Konzepten aufgebaut wurden. Diese waren Ende der Neunzigerjahren völlig neuartig und haben sich über die Jahre durch das Aufkommen von ähnlichen, in dieser Arbeit aufgezeigten, Ansätzen bestätigt und sind zur heutigen Zeit weiterhin aktuell. Die vorgestellten Konzepte für die Softwareentwicklung bestehend aus der Modellierung und Ausführung von Geschäftsprozessen kombiniert mit der Verifikation (breits zur Designzeit) und der automatischen Generierung der ausführbaren Applikation stellt eine ein-

[18] Springer Science+Business Media
[19] Java Database Connectivity

heitliche Vorgehensweise dar, die sich in Forschungs- und Industrieprojekten als erfolgreich erwiesen hat.

11.2 Ausblick

Der heutige Trend der Software-Entwicklung geht verstärkt in Richtung Modellierung und Orchestrierung von Web Services zur Erstellung von neuen Applikationen. Das Denken in Services und die Synergien von bestehenden bzw. wiederverwendbaren Diensten spiegeln den Grundgedanken der Service-orientierten Softwareentwicklung wider.

Bestehende Anwendungen von Drittanbietern ob kommerziell oder Open Source werden zu einem System orchestriert. Mittlerweile können mit der heutigen Drittanbietersoftware die damaligen proprietären Komponenten realisiert werden, hierzu gehören Lösungen von Drittanbietern für z.B. Foren, DMS [20] oder CMS [21]. Es existieren Software-Bibliotheken und APIs [22], die eine Entwicklung von Applikationen massgeblich vereinfachen. Zum Beispiel wird mit der JPA [23] [BHRS07] [mic09e] dem Entwickler eine API an die Hand gegeben, die die Applikation von der Datenbankschicht entkoppelt. Dieses musste zur Entstehungszeit des OCS aufwändig realisiert werden. Die heutige Verfügbarkeit von Architekturen, Frameworks und Sprachen wie z.B. Java EE [mic09d], .Net [Mic09b], Spring [OLW+08], JBoss Seam [JBo09], JSF [mic09g], BPEL oder LDAP [Zör08] [LU06], stellen eine große Bandbreite von direkt verwendbaren Konzepten und Technologien bereit. War noch vor einigen Jahren die eigentliche Konzipierung der Entwicklung von Software eine große Herausforderung, die die Realisierung des Gesamtpaketes bedeutete, so besteht heute die Schwierigkeit darin, die adäquaten Konzepte und die passenden Technologien für die geplante Software-Applikation aus der Palette von existierenden Lösungen zu selektieren.

Die Modellierung von Geschäftsprozessen, die beim OCS und seiner Dienstfamilie bereits vor Jahren konsequent verfolgt wurden, hat heute in den gängigen Entwicklungsumgebungen Einzug erhalten. Neue Entscheidungssysteme werden entwickelt, die existierende Technologien verwenden und auf eine breite Palette von vorhandenen Lösungen zurückgreifen können. Die Entwickler erhalten heutzutage Entwicklungswerkzeuge an die Hand, die für den Benutzer ein Gesamtpaket an Technologien zur Entwicklung von Software bereitstellen.

[20] Dokumenten-Management-System
[21] Content-Management-System: Inhaltsverwaltungssystem
[22] Application Programming Interface: Programmierschnittstelle
[23] Java Persistence API

Der OCS befindet sich seit 10 Jahren (erstmalig für TACAS [24] im Jahre 2000) erfolgreich im Einsatz für wissenschaftliche Konferenzen. Er wurde stetig weiterentwickelt und wuchs zu einem robusten und flexiblen System heran. Der OCS ist ein komplexes Begutachtungssystem geworden, in dem über die Jahre Kundenanforderungen und Benutzerrückmeldungen in Form von Features berücksichtigt wurden. Wollten die Benutzer zur Entstehungszeit des Dienstes alles selbst konfigurieren, so geht der Trend heutzutage in Richtung Einfachheit ohne aufwändige Konfigurationen vornehmen zu müssen. Die heutige Softwareentwicklung basiert auf Standards, es werden standardisierte Technologien eingesetzt und bereits existierende WebServices in die eigene Entwicklung integriert, was Entwicklung und Wartung vereinfacht.

Zur Zeit wächst eine neue Generation des OCS heran, welche die vorgestellten Konzepte dieser Arbeit mit neusten Technologien vereint. Zu den grundlegenden Konzepten gehören weiterhin der Feature-orientierte und Modellgetriebene Ansatz, mit dem die Geschäftswelt auf einfache Weise auf die Informatik, insbesondere die Softwareentwicklung, projiziert werden kann. Lebenszyklen von Geschäftsobjekten und der Geschäftsprozesse werden modelliert. Dieses basiert auf einer Abstraktionsebene, die für Anwendungsexperten ohne tiefgreifende Programmierkenntnisse verständlich und anwendbar ist. Services werden entkoppelt entwickelt und bilden durch Orchestrierung die Geschäftsprozesse ab. Es werden gezielt Standards eingesetzt, wie z.B. ein Dokumentenmanagementsystem eines Drittanbieters, was eine Wartung des eigenen Systems erleichtern soll und den Entwicklungsaufwand minimiert. Des Weiteren wird der Aspekt der Einfachheit in Bezug auf die Konfiguration und die Ergonomie der Konferenzen berücksichtigt. Informationen sollen noch kompakter dargestellt werden und Standardworkflows in Form von vordefinierten Konfigurationen, auch der vorangegangenen Konferenz, verfügbar sein. Zum Einsatz kommen Technologien wie das Web-Framework *Tapestry* [Kol08], *Apache DS* als LDAP-Server zur Authentifizierung und JPA zur Entkopplung der Datenbankschicht. Die JCR-API [25] dient dem einheitlichen und standardisierten Zugriff auf Dokumentenmanagementsysteme, wobei die Implementierung *Apache Jackrabbit* verwendet wird. Als Diskussionskomponente wird die Drittanbietersoftware jForum [Tea09b] eingesetzt, die mittels des REST [26]-Ansatzes [Fie00] angebunden wird. Als Softwarearchitektur wird auf Java EE aufgebaut.

Neben der Einführung neuer Standards und der Anpassung der Ergonomie, wäre die Erweiterung des Forum-Features durch eine integrierte Komponen-

[24] Tools and Algorithms for the Construction and Analysis of Systems
[25] Java Content Repository
[26] Representational State Transfer

te für eine Videokonferenz ein weiterer Schritt in Richtung eines *virtuellen* PC Meetings. Dieses würde den Trend heutige PC Meetings kaum noch physikalisch stattfinden zu lassen weiterhin unterstützen und den Aufwand für derartige Treffen minimieren.

Literatur

[And06] Thomas Andres. *AGILITÄT durch ARIS Geschäftsprozessmanagement*, chapter Vom Geschäftsprozess zur Anwendung: Modellgetriebene Entwicklung betriebswirtschaftlicher Software, pages 231–242. Springer Berlin Heidelberg, 2006.

[BHRS07] Robert F. Beeger, Arno Haase, Stefan Roock, and Sebastian Sanitz. *Hibernate - Persistenz in Java-Systemen mit Hibernate und der Java Persistence API*. dpunkt.verlag, 2007.

[BM06] Markus Bajohr and Tiziana Margaria. Matrics: A service-based management tool for remote intelligent configuration of systems. *ISSE*, 2(2):99–111, 2006.

[BM08] Markus Bajohr and Tiziana Margaria. High service availability in matrics for the ocs. In *ISoLA*, pages 572–586, 2008.

[BM09] Alexandre Bragança and Ricardo Machado. A model-driven approach for the derivation of architectural requirements of software product lines. *Innovations in Systems and Software Engineering*, 5(1):65–78, March 2009.

[BMRS06] Marco Bakera, Tiziana Margaria, Clemens Renner, and Bernhard Steffen. Game-based model checking for reliable autonomy. In *SMC-IT'06, 2nd IEEE Int. Conf. on Space Mission Challenges for Information Technology, Workshop on Autonomous and Autonomic Systems*. IEEE Computer Society, 2006.

[BMRS07a] M. Bakera, T. Margaria, C. Renner, and B. Steffen. Property-driven functional healing: Playing against undesired behavior. In *10th CONQUEST*, 2007.

[BMRS07b] M. Bakera, T. Margaria, C. Renner, and B. Steffen. Verification, diagnosis and adaptation: Tool supported enhancement of

the model-driven verification process. In *Wrksh. on Formal Methods in Avionics, Space and Transport (ISOLA)*, pages 85–98. Revue des Nouvelles Technologies de l'Information (RNTI-SM-1), 2007.

[BMRS08] Marco Bakera, Tiziana Margaria, Clemens D. Renner, and Bernhard Steffen. Model Checking with the JavaABC Framework - From METAGame to GEAR. In *QEES'08, Intern. Symposium on Quality Engineering for Embedded Systems - In conjunction with the ECMDA 2008*. IEEE CS, 2008.

[BMS06] Markus Bajohr, Tiziana Margaria, and Bernhard Steffen. Service based enabling service availability in the matrics: A model-driven approach. In *ISoLA*, pages 317–324, 2006.

[BNB+06] Siddharth Bajaj, Nataraj Nagaratnam, Don Box, Dave Chappell, and Francisco Curbera. Web services policy 1.2 - framework (ws-policy). Technical report, W3C, April 2006.

[Bra01] Volker Braun. *A Coarse-granular Approach to Software Development allowing Non-Programmers to Build and Deploy Reliable, Web-based Applications*. PhD thesis, Dissertation, Technische Universität Dortmund, Lehrstuhl für Programmiersysteme, Dortmund, Deutschland, 2001.

[CBBa06] Francisco Curbera, Keith Ballinger, Bobby Bissett, and Don Box and. Web services metadata exchange (ws-metadataexchange). Technical report, BEA Systems Inc., Computer Associates International, Inc., International Business Machines Corporation, Microsoft Corporation, Inc., SAP AG, Sun Microsystems, webMethods, August 2006.

[CGP01] Emund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 2001.

[Cza04] Krzysztof Czarnecki. Overview of generative software development. In *UPP*, pages 326–341, 2004.

[DDB+04] Michael Dimov, Minye Dong, Youssef Belkhadir, Michael Drazek, Jaouad El Jerroudi, Christian Kuete-Tsatedem, Daniel Pierlings, Mohammed Said, Dennis Saßmannshausen, Holger Willebrandt, and Jing Zhou. Eureka 1 -

entscheidungs-unterstützung: Rollengerecht, effizient, kooperativ, allgegenwärtig. Technical report, PG EUREKA I - Technische Universität Dortmund, Lehrstuhl für Programmiersysteme, September 2004.

[DGSZ94] Guido Dinkhoff, Volker Gruhn, Armin Sallmann, and Michael Zielonka. Business process modelling in the workflow-management environment leu. In *ER '94: Proceedings of the13th International Conference on the Entity-Relationship Approach*, pages 46–63, London, UK, 1994. Springer-Verlag.

[DLHPB09] Giusy Di Lorenzo, Hakim Hacid, Hye-young Paik, and Boualem Benatallah. Data integration in mashups. *SIGMOD Rec.*, 38(1):59–66, 2009.

[EG07] Robert J. Ennals and Minos N. Garofalakis. Mashmaker: mashups for the masses. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1116–1118, New York, NY, USA, 2007. ACM.

[EJS93] E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. On model-checking for fragments of mu-calculus. In *CAV '93: Proceedings of the 5th International Conference on Computer Aided Verification*, pages 385–396, London, UK, 1993. Springer-Verlag.

[Erl04] Thomas Erl. *Service-Oriented Architecture*. Prentice Hall, 2004.

[FAAD07] Mark Ford, Ashish Agrawal, Mike Amend, and Manoj Das. Web services human task (ws-humantask), version 1.0. Technical report, Active Endpoints Inc., Adobe Systems Inc., BEA Systems Inc., International Business Machines Corporation, Oracle Inc., SAP AG., June 2007.

[Fie00] Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, 2000. Chair-Taylor an, Richard N.

[FK92] David Ferraiolo and Richard Kuhn. Role-based access control. In *In 15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.

[Fow03] Martin Fowler. *UML konzentriert*. Addison Wesley Verlag, 2003.

[GDLHBH05] Martin Gudgin, Giovanni Della-Libera, Phillip Hallam-Baker, and Maryann Hondo. Web services security policy language (ws-securitypolicy) version 1.1. Technical report, International Business Machines Corporation, Microsoft Corporation, RSA Security Inc., VeriSign Inc, July 2005.

[GKA+07] Yaron Goland, Neelakantan Kartha, Alexandre Alves, Assaf Arkin, Sid Askary, Ben Bloch, Francisco Curbera, Sterling, Dieter König, Vinkesh Mehta, Satish Thatte, Danny van der Rijn, Prasad Yendluri, and Alex Yiu. Web services business process execution language version 2.0. Technical report, WS-BPEL TC OASIS, April 2007.

[Gri04] P. Griffin. Introduction to xacml. Technical report, Bea Systems, February 2004.

[Gso04] Claudia Gsottberger. *Ein Framework zur modularisierten und pattern-basierten Entwicklung von zuverlässigen, personalisierten, web-basierten Applikationen.* PhD thesis, Dissertation, Technische Universität Dortmund, Lehrstuhl für Programmiersysteme, Dortmund, Deutschland, 2004.

[HB08] Graham Hughes and Tevfik Bultan. Automated verification of access control policies using a sat solver. *Int. J. Softw. Tools Technol. Transf.*, 10(6):503–520, 2008.

[HBTE+05] Abdelazziz Ait Ben Haddou, Dominik Brink, Omar Zaouaoui Elidrissi Taieb Elfakiri, Mikhail Farnassov, Holger Hopf, Markus Korte, Philipp Langer, Ganna Ostapchuk, Alexander Powolozki, Abdelmounaim Ramadane, and Marc von Renteln. Eureka 2 - entscheidungs-unterstützung: Rollengerecht, effizient, kooperativ, allgegenwärtig. Technical report, PG EUREKA II - Technische Universität Dortmund, Lehrstuhl für Programmiersysteme, July 2005.

[HG98] Otthein Herzog and Andreas Günter, editors. *KI-98: Advances in Artificial Intelligence, 22nd Annual German Conference on Artificial Intelligence, Bremen, Germany, September 15-17, 1998, Proceedings*, volume 1504 of *Lecture Notes in Computer Science*. Springer, 1998.

[HMM+06] Martina Hörmann, Tiziana Margaria, Thomas Mender, Ralf Nagel, Michael Schuster, Bernhard Steffen, and Hong Trinh. The

jabc appraoch to collaborative development of embedded applications. In *CCE'06, Int. Worksh. on Challenges in Collaborative Engineering - State of the Art and Future Challenges on Collaborative Design*, 2006.

[Hol06] Holger Willebrandt. Service-orientierte Entwicklung einer Applikation zur Konfiguration und Erzeugung von Tagungsbänden. Master's thesis, Diplomarbeit, Technische Universität Dortmund, Lehrstuhl für Programmiersysteme, Dortmund, Deutschland, November 2006.

[HW08] Stefan Huth and Thomas Wieland. Geschäftsprozessmodellierung mittels software-services auf basis der epk. In *Service-orientierte Architekturen*, pages 61–76. Springer-Verlag, 2008.

[IT92] ITU-T. Recommendation Q.1203. "Intelligent Network - Global Functional Plane Architecture". Technical report, Standardization Sector of ITU, October 1992.

[ITU93] ITU. General recommendations on telephone switching and signaling - intelligent network: Introduction to intelligent network capability set 1, Recommendation Q.1211. Technical report, Standardization Sector of ITU, Geneva, March 1993.

[JMS06] Sven Jörges, Tiziana Margaria, and Bernhard Steffen. Formulabuilder: A tool for graph-based modelling and generation of formulae. In *Proc. ICSE '06*, pages 815–818. ACM Press, 2006.

[KCH+90] Kyo Kang, Sholom Cohen, James Hess, William Novak, and A. S. Peterson. Feature-oriented domain analysis (foda) feasibility study. Technical report, Software Engineering Institute, Carnegie Mellon University Pittsburgh, Pennsylvania 1521, 1990.

[KJMS09] Christian Kubczak, Sven Jörges, Tiziana Margaria, and Bernhard Steffen. extreme model-driven design with jabc. In *R. Vogel (ed.) Proc. of the Tools and Consultancy Track of the Fifth European Conference on Model-Driven Architecture Foundations and Applications (ECMDA-FA)*, volume WP09-12 of CTIT proceedings, pages 78–99, P.O. Box 217, 7500 AE Enschede, The Netherlands, University of Twente, Centre for Telematics and Information Technology (CTIT), 2009.

[KKL+05] Matthias Kloppmann, Dieter König, Frank Leymann, Gerhard Pfau, and Alan Rickayzen. Bpel-spe - ws-bpel extension for subprocesses. Technical report, IBM and SAP, September 2005.

[KM05] Martin Karusseit and Tiziana Margaria. Feature-based Modelling of a Complex, Online-Reconfigurable Decision Support Service. In *WWV '05, 1st Int. Worksh. Automated Specif. and Verification of Web Sites*, March 2005. ENTCS 1132.

[KM06] Martin Karusseit and Tiziana Margaria. A web-based runtime-reconfigurable role management service. In *WWV '06: Proceedings of the 2nd Int'l. Workshop on Automated Specification and Verification of Web Systems*, pages 53–60, Washington, DC, USA, 2006. IEEE Computer Society.

[KMW08] Martin Karusseit, Tiziana Margaria, and Holger Willebrandt. Policy expression and checking in xacml, ws-policies, and the jabc. In *TAV-WEB '08: Proceedings of the 2008 workshop on Testing, analysis, and verification of web services and applications*, pages 20–26, New York, NY, USA, 2008. ACM.

[Kol08] Alexander Kolesnikov. *Tapestry 5 - Building Web Applications*. Packt Publishing, 2008.

[LK04] Daniel R. Licata and Shriram Krishnamurthi. Verifying interactive web programs. In *ASE '04: Proceedings of the 19th IEEE international conference on Automated software engineering*, pages 164–173, Washington, DC, USA, 2004. IEEE Computer Society.

[LMS01] B. Lindner, T. Margaria, and B. Steffen. Ein personalisierter internetdienst für wissenschaftliche begutachtungsprozesse. *GI-VOI-BITKOM-OCG-TeleTrusT Konferenz Elektronische Geschäftsprozesse (eBusiness Processes)*, 2001.

[LU06] Oliver Liebel and John Martin Ungar. *OpenLDAP*. Galileo Press, 2006.

[Mar05] T. Margaria. Web services-based tool-integration in the eti platform. *SoSyM*, 4(2):141–156, 2005.

[Mar07] Tiziana Margaria. Service is in the eyes of the beholder. *IEEE Computer - Innivative Technology for Computer Professionals*, 40(11):33–37, 2007.

[MBK97] T. Margaria, V. Braun, and J. Kreileder. Interacting with eti: A user session. *STTT*, 1(1-2):49–63, 1997.

[MHBK03] Hiroshi Maruyama, Maryann Hondo, Don Box, and Chris Kaler. Web services policy assertions language (ws-policyassertions) version 1.1. Technical report, BEA Systems, Inc, International Business Machines Corporation, Microsoft Corporation, SAP AG, May 2003.

[MK02] Tiziana Margaria and Martin Karusseit. Community usage of the online conference service: an experience report from three cs conferences. In *I3E '02: Proceedings of the IFIP Conference on Towards The Knowledge Society*, pages 497–511, Deventer, The Netherlands, The Netherlands, 2002. Kluwer, B.V.

[Mos05] Tim Moses. extensible access control markup language (xacml) - version2.0. Technical report, OASIS, Februar 2005.

[MOSS99] Markus Müller-Olm, David .A. Schmidt, and Bernhard Steffen. Model-Checking: A Tutorial Introduction. *Proc. SAS*, pages 330–354, September 1999.

[MRSL07] T. Margaria, H. Raffelt, B. Steffen, and M. Leucker. The learnlib in fmics-jeti. In *Proc. ICECCS 2007, 12th IEEE Int. Conf. on Engineering of Complex Computer Systems*, number 2 in Proc. ICECCS, Auckland (NZ), July 2007. IEEE Computer Soc. Press.

[MS04a] Tiziana Margaria and Bernhard Steffen. Aggressive model driven development for the management of service evolution. In *In Annual Review of Communication, Int. Engineering Consortium, IEC*, volume 57, 2004.

[MS04b] Tiziana Margaria and Bernhard Steffen. Lightweight coarse-grained coordination: a scalable system-level approach. *Int. J. Softw. Tools Technol. Transf.*, 5(2):107–123, 2004.

[MS06] Tiziana Margaria and Bernhard Steffen. Service Engineering: Linking Business and IT. *IEEE Computer - Innovative Technology for Computer Professionals*, 39(10):45–55, 2006.

[MS08] T. Margaria and B. Steffen. Agile IT: Thinking in User-Centric Models. In *Proc. ISoLA 2008*, CCIS N.17, pages 493–505. Springer, 2008.

[MS09] Tiziana Margaria and Bernhard Steffen. *In Handbook of Research on Business Process Modeling*, chapter Business Process Modelling in the jABC: The One-Thing Approach. IGI Global, 2009.

[MSR05] Tiziana Margaria, Bernhard Steffen, and Manfred Reitenspieß. Service-Oriented Design: The Roots. In *ICSOC 2005: 3rd ACMSIGSOFT/SIGWEB Int. Conf. on Service-Oriented Computing*, LNCS N.3826, pages 450–464, Amsterdam, December 2005. Springer Verlag.

[Nag09] Ralf Nagel. *Technische Herausforderungen modellgetriebener Beherrschung von Prozesslebenszyklen aus der Fachperspektive: Von der Anforderungsanalyse zur Realisierung*. PhD thesis, Dissertation, Technische Universität Dortmund, Lehrstuhl für Programmiersysteme, Dortmund, Deutschland, 2009.

[Nie00] Oscar Nierstrasz. Identify the champion. In N. Harrison, B. Foote, and H. Rohnert, editors, *Pattern Languages of Program Design*, volume 4, pages 539–556. Addison Wesley, 2000.

[Nie03] Oliver Niese. *An Integrated Approach to Testing Complex Systems*. PhD thesis, Dissertation, Technische Universität Dortmund, Lehrstuhl für Programmiersysteme, Dortmund, Deutschland, 2003.

[NMH+01] Oliver Niese, Tiziana Margaria, Andreas Hagerer, Markus Nagelmann, Bernhard Steffen, Georg Brune, and Hans dieter Ide. An automated testing environment for CTI systems using concepts for specification and verification of workflows. In *Annual Review of Communication*, volume 54, pages 927–936. Int. Engineering Consortium, Chicago (USA), 2001.

[NSM+01] Oliver Niese, Bernhard Steffen, Tiziana Margaria, Andreas Hagerer, Georg Brune, and Hans-Dieter Ide. Library-based design and consistency checking of system-level industrial test cases. In *Fundamental Approaches to Software Engineering*, volume 2029/2001, pages 233–248. Springer Verlag, 2001.

[OCS08] OCS-Team. OCS Introduction, 2008. Lehrstuhl für Programmiersysteme, TU Dortmund (Hrsg.).

[OLW+08] Richard Oates, Thomas Langer, Stefan Wille, Torsten Lueckow, and Gerald Bachlmayr. *Spring & Hibernate*. Carl Hanser Verlag, 2008.

[SAM+08] David E. Simmen, Mehmet Altinel, Volker Markl, Sriram Padmanabhan, and Ashutosh Singh. Damia: data mashups for intranet applications. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1171–1182, New York, NY, USA, 2008. ACM.

[SCFY96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[Sei02] Heinrich Seidlmeier. *Prozessmodellierung mit ARIS*. vieweg, Wiesbaden 2002.

[SJWM09] Bernhard Steffen, Sven Jörges, Christian Wagner, and Tiziana Margaria. Maintenance, or the 3rd dimension of extreme model-driven design. *ICSM 2009 - 25th IEEE International Conference on Software Maintenance, to appear*, 2009.

[SM99] Bernhard Steffen and Tiziana Margaria. METAFrame in Practice: Design of Intelligent Network Services. In *Correct System Design - Recent Insights and Advances*, LNCS N.1710, pages 390–415. Springer-Verlag, Heidelberg, Germany, October 1999.

[SMCB96] Bernhard Steffen, Tiziana Margaria, Andreas Claßen, and Volker Braun. The metaframe'95 environment. In *Proc. CAV'96, LNCS*, pages 450–453. Springer, 1996.

[SMN+07] Bernhard Steffen, Tiziana Margaria, Ralf Nagel, Sven Jörges, and Christian Kubczak. Model-driven development with the jabc. *Lecture Notes in Computer Science*, 4383:92–108, May 2007.

[SN07] Bernhard Steffen and Prakash Narayan. Full life-cycle support for end-to-end processes. *Computer*, 40(11):64–73, 2007.

[Sta01] Richard Van De Stadt. Cyberchair: A web-based groupware application to facilitate the paper reviewing process. available at www.cyberchair.org, 2001.

[Ste91] Bernhard Steffen. Data Flow Analysis as Model Checking. In *TACS '91: Proceedings of the International Conference on Theoretical Aspects of Computer Software*, pages 346–365. Springer-Verlag, 1991.

[SVC06] Thomas Stahl, Markus Voelter, and Krzysztof Czarnecki. *Model-Driven Software Development: Technology, Engineering, Management*. John Wiley & Sons, 2006.

[Tay08] Camillo J. Taylor. On the optimal assignment of conference papers to reviewers. Technical report, No. MS-CIS-08-30, University of Pennsylvania Department of Computer and Information Science, January 2008.

[WCL+05] Sanjiva Weerawarana, Francisco Curbera, Frank Leymann, Tony Storey, and Donald F. Ferguson. *Web Services Platform Architecture : SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More*. Prentice Hall PTR, March 2005.

[Web99] Herbert Weber. Continuous engineering of information and communication infrastructures (extended abstract). In *FASE '99: Proceedings of the Second Internationsl Conference on Fundamental Approaches to Software Engineering*, pages 22–29, London, UK, 1999. Springer-Verlag.

[WM08] Stephen A. White and Derek Miers. *BPMN Modeling and Reference Guide*. Future Strategies Inc., Lighthouse Pt, FL, City, 2008.

[Zör08] Stefan Zörner. *LDAP für Java-Entwickler*. entwickler.press, 2008.

[ZRN08] Nan Zang, Mary Beth Rosson, and Vincent Nasser. Mashups: who? what? why? In *CHI '08: CHI '08 extended abstracts on Human factors in computing systems*, pages 3171–3176, New York, NY, USA, 2008. ACM.

Web Referenzen

[All09] OpenAjax Alliance. *Asynchronous JavaScript and XML*, July 2009. http://de.wikipedia.org/wiki/Ajax_(Programmierung).

[Apa09] Apache. *Apache Velocity Project*, July 2009. http://velocity.apache.org/.

[Ari09] Aries. *Editorial Manager*, July 2009. www.editorialmanager.de.

[Ber09] Michel Berkelaar. *lp_solve*, July 2009. http://lpsolve.sourceforge.net.

[Com09] Open Source Community. *PostgreSQL*, July 2009. http://www.postgresql.org/.

[Con09] W3C World Wide Web Consortium. *Die Homepage des W3C*, July 2009. http://www.w3.org/.

[Fou09a] Apache Software Foundation. *Apache ActiveMQ*, July 2009. http://activemq.apache.org/.

[Fou09b] Apache Software Foundation. *Apache HTTP Server*, July 2009. http://httpd.apache.org.

[Fou09c] Apache Software Foundation. *Apache Tomcat*, July 2009. http://tomcat.apache.org.

[GG09] Rich Gerber and Paolo Gai. *START V2 Conference Manager*, July 2009. http://www.softconf.com.

[Goo09] Google. *GWT - Google Web Toolkit*, July 2009. http://code.google.com/webtoolkit/.

[Gro09] Zakon Group. *OpenConf*, July 2009. http://www.openconf.com.

[Hat09] Red Hat. *JBoss Applikation Server*, July 2009. http://www.jboss.com.

[Hei09a] Springer Verlag Heidelberg. *LNCS Transactions Subline*, July 2009. http://www.springer.com/computer/lncs?SGWID=0-164-6-546809-0.

[Hei09b] Springer Verlag Heidelberg. *OCS als Springer Konferenzsystem*, July 2009. http://www.springer.com/computer/lncs?SGWID=0-164-6-447109-0.

[Hei09c] Springer Verlag Heidelberg. *Springer Homepage*, July 2009. http://www.springer.com.

[Hei09d] Springer Verlag Heidelberg. *Springer LNCS Homepage*, July 2009. http://www.springer.com/computer/ lncs?SGWID=0-164-12-72397-0.

[INC09] SUN MICROSYSTEMS INC. *The Java Programming Language*, July 2009. http://java.sun.com.

[JBo09] JBoss. *JBoss Seam*, July 2009. http://www.jboss.com/products/seam/.

[Kri09] Shriram Krishnamurthi. *Continue*, July 2009. http://continue2.cs.brown.edu/.

[Mic09a] Microsoft. *Microsoft Popfly*, 2009. http://www.popfly.com.

[Mic09b] Microsoft. *.Net Framework*, July 2009. http://www.microsoft.com/NET/.

[Mic09c] Microsoft. *XLANG*, July 2009. http://msdn.microsoft.com/en-us/library/aa577463(BTS.10).aspx.

[mic09d] Sun microsystems. *Java EE*, July 2009. http://java.sun.com/javaee/.

[mic09e] Sun microsystems. *Java Persistence API*, July 2009. http://java.sun.com/javaee/technologies/persistence.jsp.

[mic09f] SUN microsystems. *Java Servlet Technology*, July 2009. http://java.sun.com/products/servlet/.

[mic09g] Sun microsystems. *JavaServer Faces*, July 2009. http://java.sun.com/javaee/javaserverfaces/.

[mic09h] SUN microsystems. *JAXB - Java API for XML Processing*, July 2009. http://java.sun.com/webservices/technologies/.

[mic09i] SUN microsystems. *JAXB Article*, July 2009. http://java.sun.com/developer/technicalArticles/WebServices/jaxb/.

[mic09j] SUN microsystems. *JDBC - Java Database Connectivity*, July 2009. http://java.sun.com/javase/technologies/database/.

[mic09k] SUN microsystems. *JMS - Java Message Service*, July 2009. http://java.sun.com/products/jms/.

[mic09l] SUN microsystems. *SUN Homepage*, July 2009. www.sun.com.

[OAS09] OASIS. *OASIS Homepage*, July 2009. http://www.oasis-open.org/home/index.php.

[Ora09] Oracle. *Database*, July 2009. http://www.oracle.com/database.

[oT09] oAW Team. *openArchitectureWare framework*, 2009. http://www.openarchitectureware.org/.

[Pre09a] Thomas Preuss. *Confman*, July 2009. http://www.ifi.uio.no/confman/ABOUT-ConfMan/.

[Pre09b] Thomas Preuss. *ConfMaster*, July 2009. http://www.confmaster.net.

[Pre09c] Thomas Preuss. *Coniant: ConfMaster*, July 2009. http://www.coniant.net.

[Pro09] Public Knowledge Project. *Open Conference Systems*, July 2009. http://pkp.sfu.ca/ocs.

[Sch09] Henning Schulzrinne. *Editorial's Assistent: EDAS*, July 2009. http://www.edas.info.

[Sno09] R. Snodgrass. *Summary of conference management software*, July 2009. http://www.acm.org/sigs/sgb/summary.html.

[Sof09] RedWhale Software. *Conference Reviewing System: CRS*, July 2009. http://www.conferencereview.com.

[Sys09] Open Conference/Journal System. *Public Knowledge Project*, July 2009. http://pkp.sfu.ca, 2009.

[Tea09a] AndroMDA Team. *AndroMDA Gode-Generator Frameword*, 2009. http://www.andromda.org/.

[Tea09b] JForum Team. *JForum*, July 2009. http://www.jforum.net/.

[vdS09a] Richard van de Stadt. *CyberChair*, July 2009. www.borbala.com/cyberchair/.

[vdS09b] Richard van de Stadt. *CyberChairPRO*, July 2009. http://www.borbala.com.

[Vor09] Andrei Voronkov. *EasyChair*, July 2009. http://www.easychair.org.

[Wes09] Ajamu Wesley. *Web Services Flow Language (WSFL)*, July 2009. http://www.ibm.com/developerworks/webservices/library/ws-wsfl1/.

[WML09] WML. *Website Meta Language*, July 2009. http://thewml.org/.

[Yah09] Yahoo. *Yahoo Pipes*, 2009. http://pipes.yahoo.com.