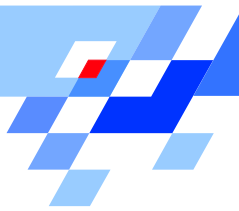
UNIVERSITÄT DORTMUND

Fachbereich Informatik
Lehrstuhl für Software-Technologie



M E M O Nr. 131

Endbericht der PG nightshift: Dokumentation der verteilten Geschäftsprozesse im FBI und Umsetzung von Teilen dieser Prozesse im Rahmen eines FBI-Intranets basierend auf WAP- und Java-Technologie

Yalda Ariana, Oliver Effner, Marcel Gleis, Martin Krzysiak,
Jens Lauert, Thomas Louis, Carsten Röttgers, Kai Schwaighofer,
Martin Testrot, Uwe Ulrich, Xingguang Yuan

Prof. Dr. Volker Gruhn, Sami Beydeda

Februar 2003

Internes Memorandum des
Lehrstuhls für Software-Technologie
Prof. Dr. Ernst-Erich Doberkat
Fachbereich Informatik
Universität Dortmund
Baroper Straße 301

D-44227 Dortmund

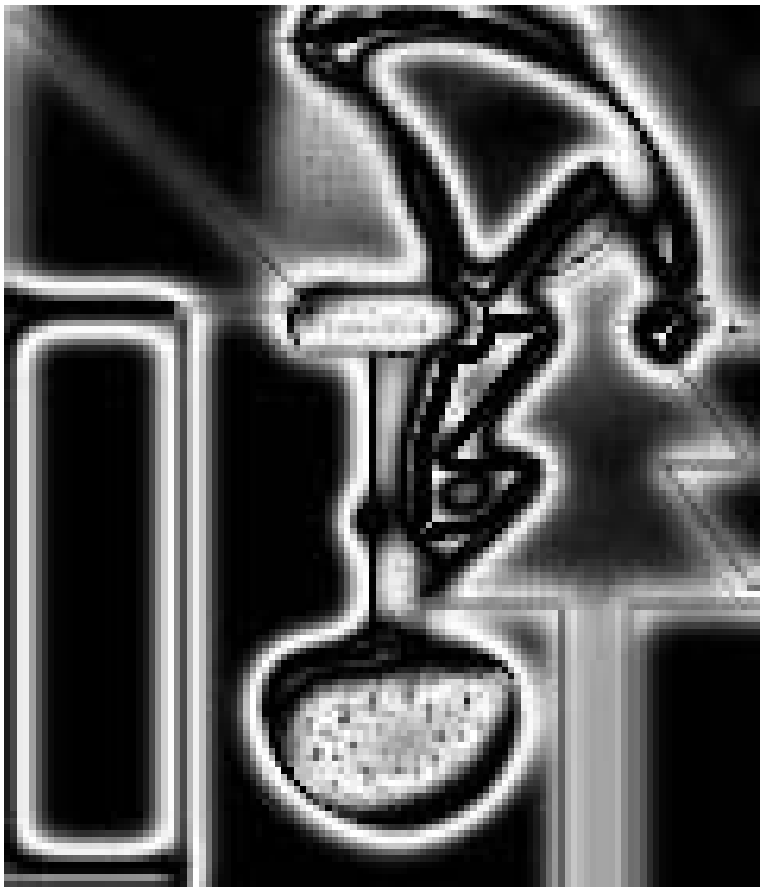
ISSN 0933-7725

ENDBERICHT

PG nightshift

Dokumentation der verteilten Geschäftsprozesse im FBI und Umsetzung von Teilen dieser Prozesse im Rahmen eines FBI-Intranets basierend auf WAP- und Java-Technologie

WS 2001/2002 bis SS 2002



Yalda Ariana, Oliver Effner, Marcel Gleis, Martin Krzysiak, Jens Lauert, Thomas Louis, Carsten Röttgers, Kai Schwaighofer, Martin Testrot, Uwe Ulrich, Xingguang Yuan

Betreuer: Prof. Dr. Volker Gruhn, Sami Beydeda

Lehrstuhl Software-Technologie
Fachbereich Informatik
Universität Dortmund
http://ls10-www.cs.uni-dortmund.de/LS10/Pages/pgs.shtml

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Übersicht über die Kapitel

Jedes Kapitel beschreibt einen Teilaspekt der gesamten PG. Die thematische Gliederung der Kapitel entspricht weitestgehend dem chronologischen Ablauf der einzelnen PG -Phasen. Da diese Phasen nicht immer isoliert behandeln werden können, gibt es an einigen Stellen Überschneidungen (inhaltlich und zeitlich). Generell werden aber schon durch die Anordnung der Kapitel der Ablauf und das Vorgehen der PG wiedergegeben. Dieser Bericht besteht aus den folgenden Kapiteln:

- **Einleitung**
  In der Einleitung werden die Umstände und Ursachen skizziert, aus denen die Motivation zu dieser PG hervorging.

- **PG Organisation**
  Dieses Kapitel beinhaltet die organisatorischen Eckdaten zur PG (Name, Umfang, etc.) und eine Abgrenzung der Minimalziele.

- **Technische Grundlagen**
  Hier werden die Grundtechnologien, die für die Umsetzung der Ziele diskutiert wurden, vorgestellt. Diese Kapitel entsprechen inhaltlich der Seminarveranstaltung zu Beginn der PG. Einige dieser Technologien sind nur kurz der Vollständigkeit halber erwähnt, z.B. UMTS, weil sie für die Umsetzung der PG -Ziele keine Berücksichtigung fanden.

- **Projektmanagement**
  Die einzelnen Planungsschritte und das Vorgehen bei der Projektplanung werden hier dargestellt.

- **Qualitätsmanagement**
  Dieses Kapitel behandelt die Maßnahmen, die ergriffen wurden, um die Teilziele, vor allem die der Implementierungsphase, fest- und umzusetzen.

- **Systementwurf**
  An dieser Stelle werden die Schritte beschrieben, die notwendig waren um den Entwurf des Clevery-Systems zu erstellen.

- **Systemimplementierung**
  Dieses Kapitel handelt, im Gegensatz zu Kapitel 4, von den tatsächlich eingesetzten Werkzeugen und Technologien um das Clevery-System zu implementieren. Auch eine Bewertung der einzelnen Mittel ist Gegenstand dieses Kapitels und daher sehr nützlich, falls nochmals ein derartig konzipiertes System entwickelt werden soll.

- **Clevery Anwendung**
  In diesem Abschnitt werden die Systemanforderungen und Konfigurationen behandelt, die für den Einsatz von Clevery erforderlich sind.

- **Bewertung**
  Gegenstand dieses Kapitels ist eine Bewertung der Leistungen, die während der PG erbracht wurden. Dabei stehen die in Kapitel 3 beschriebenen Minimalziele im Vordergrund.

- **Fazit**
  Abschließend wird hier ein Resümee über die ganze PG gezogen.

2 Einleitung

In der letzten Zeit wird vor allem von Seiten der Industrie immer wieder gefordert, die Informatikausbildung an den Hochschulen zu beschleunigen, um die Nachfrage an qualifiziertem IT-Personal decken zu können. In der Tat liegt die durchschnittliche Studiendauer an der Universität Dortmund bei knapp 15 Semestern, was die Notwendigkeit von Maßnahmen zur Optimierung des Informatikstudiums verdeutlicht. Allerdings wurde aus dieser Tatsache häufig gefolgert, dass die Inhalte des Studiums zu straffen seien, um die durchschnittliche Studiendauer zu senken. Auch wenn das eine mögliche Maßnahme ist, muss betont werden, dass komplizierte und ineffiziente Abläufe am Fachbereich das Studium ebenfalls in unnötiger Weise verlängern. Als Beispiel soll hier die Anmeldung einer mündlichen Prüfung dienen.

Studierende müssen zur Anmeldung einer mündlichen Prüfung folgende Schritte unternehmen:

Termin vereinbaren

Prüfer und Studierende müssen unter Beachtung von Restriktionen einen Prüfungstermin bestimmen. Hierbei muss eine Frist von zwei Wochen zwischen Anmeldung und Prüfung berücksichtigt werden. Außerdem muss dieser Termin gegebenenfalls mit einem zweiten Prüfer abgestimmt werden.

Anmeldeformular vom Prüfungsamt abholen

Als nächstes muss der Studierende ein Formular, das zum Anmelden der Prüfung benötigt wird, vom Prüfungsamt abholen und ausfüllen.

Unterlagen vom Prüfer unterzeichnen lassen

Das ausgefüllte Formular muss beim Prüfer zur Unterschrift vorgelegt werden. Hierbei sind die Sprechzeiten des Prüfers zu beachten.

Unterlagen beim Prüfungsamt abgeben

Der letzte Schritt bei der Anmeldung zu einer mündlichen Prüfung besteht aus dem Abgeben des unterschriebenen Anmeldeformulars beim Prüfungsamt. Im einfachsten Fall wird das Formular in den Briefkasten des Prüfungsamtes eingeworfen.

In diesem Beispiel können mögliche Verzögerungen, die bereits die Anmeldephase unnötig verlängern und somit unnötigen Zeitverlust für den Studierenden zur Folge haben, vor allem aus zwei Gründen entstehen. Zum einen ist der Anmeldeprozess ineffizient. Der Studierende ist dafür verantwortlich, dass das Anmeldeformular vom Prüfungsamt zum Prüfer gelangt, vom Prüfer unterzeichnet wird und wieder beim Prüfungsamt vorliegt. Diese Schritte könnten aber auch, solange nicht Bestimmungen der Prüfungsordnung des Fachbereiches verletzt werden, direkt zwischen Prüfungsamt und Prüfer abgewickelt werden.

Zum anderen fehlen Technologien, die den Prozess der Prüfungsanmeldung unterstützen und ihn effizienter machen. Das Vereinbaren des Prüfungstermins könnte auch programmunterstützt über das Internet geschehen. Der Studierende könnte Terminwünsche online über das Internet eingeben und diese könnten mit der entsprechenden Anwendung auf der Seite des Prüfers validiert und vergeben werden. Durch die zusätzliche Bereitstellung eines geeigneten Zugangs für beispielsweise WAP-Mobiltelefone oder PDAs wäre der Studierende nicht nur auf einen Computerzugang angewiesen und könnte die Anmeldung prinzipiell von überall erledigen, d.h. vor allem ohne persönlich innerhalb eines kurzen Zeitfensters an der Hochschule zu sein.

Bereits die in diesem Beispiel aufgeführten Ineffizienzen tragen nicht nur zu einer Verlängerung der durchschnittlichen Studienzeiten bei. Sie führen dazu, dass Professoren und Mitarbeiter durch organisatorische Aufgaben belastet werden und weniger Kapazität für fachliche und inhaltliche Aufgaben in Forschung und Lehre übrig bleibt.

Diese Projektgruppe soll entsprechend den oben genannten Gründen zwei Ziele verfolgen. Das erste Ziel wird die Dokumentation und die mögliche Optimierung der Prozesse am Fachbereich sein. Das zweite Ziel ist die softwaretechnische Umsetzung eines Systems zur Unterstützung der optimierten Prozesse.

Die softwaretechnische Umsetzung wird dabei auf mobile Technologien ausgerichtet sein. Es soll ein System konzipiert und entwickelt werden, das den beteiligten Personen eine Interaktion mittels mobiler Endgeräte ermöglicht. Hier soll vor allem ein Zugang mit WAP-fähigen Mobiltelefonen möglich sein, wobei der Zugang mit PDAs je nach Fortschritt der Projektgruppe ebenfalls entwickelt werden soll. Das zu realisierende Softwaresystem soll plattformübergreifend lauffähig sein und mit minimalem Aufwand installierbar sein. Die Realisierung soll mit Hilfe des Komponentenmodells „Enterprise JavaBeans“ geschehen.

3 PG Organisation

3.1 PG-Titel

Dokumentation der verteilten Geschäftsprozesse im FBI und Umsetzung von Teilen dieser Prozesse im Rahmen eines FBI-Intranets basierend auf WAP- und Java-Technologie.

3.2 PG-Zeitraum

WS 2001/2002 und SS 2002

3.3 PG-Umfang

Jeweils 8 SWS

3.4 PG-Teilnehmer

Studenten:
Yalda Ariana, Oliver Effner, Marcel Gleis, Martin Krzysiak, Jens Lauert, Thomas Louis, Carsten Röttgers, Kai Schwaighofer, Martin Testrot, Uwe Ulrich, Xingguang Yuan

Betreuung:
Prof. Dr. Volker Gruhn, LS X
Sami Beydeda, LS X

3.5 PG-Aufgabe

Teile der Prozesse am Fachbereich, wie die Anmeldung einer Prüfung oder die Organisation einer Übung, sollen modelliert und nach Möglichkeit optimiert werden. Anschließend soll ein intranetbasiertes System entwickelt werden, das die Angehörigen des Fachbereichs - Studierende wie auch Lehrende - bei der Durchführung der optimierten Prozesse unterstützt. Das entwickelte System soll insbesondere mit mobilen Endgeräten wie WAP-fähigen Mobilfunktelefonen bedienbar sein. Beispielsweise soll das System den Studierenden die Anmeldung zur Teilnahme an Übungsgruppen via WAP-Mobilfunktelefone ermöglichen.

3.6 PG-Minimalziele

1. Dokumentation ausgesuchter Prozesse im Fachbereich
2. Anforderungsanalyse eines Systems zur Unterstützung der Prozesse
3. Grob- und Feinentwurf des Systems
4. Implementierung eines Systems, das mindestens zwei Prozesse am Fachbereich unterstützt und ein Interface für die beteiligten Personen über WAP-fähige Mobilfunktelefone zur Verfügung stellt

3.7 PG-Name „PG nightshift“

Der Name „PG nightshift“ entstand während der Seminarphase im Tagungshaus in Nordhelle. Die Teilnehmer der PG entschieden sich für eine Fortsetzung der Seminarvorträge bis spät in die Nacht hinein anstelle eines gemütlichen Abends in den Räumlichkeiten der Tagungsstätte. Aus dieser Entscheidung ging der Name „nightshift“ hervor. Die Vorträge wurden bis fast 4:00 Uhr morgens gehalten und ausgiebig diskutiert.

3.8 Projektname „Clevery“

Einer der wichtigsten Aspekte des Projektes ist die Ortsunabhängigkeit. Diese entlastet alle Beteiligten der Geschäftsprozesse und ermöglicht eine stressfreie Organisation. Um diesen wichtigen Punkt zu unterstreichen, wurde der Name „Clevery“ als Kombination der Begriffe „**cl**ose“ und „**every**where“ gewählt. Dies betont den Anspruch des Systems nach Ortsunabhängigkeit und suggeriert, dass der Benutzer überall „nah dran ist“. Die Assoziation des Namens „Clevery“ mit dem Wort „clever“ unterstreicht zusätzlich, dass es sich bei dem System um eine intelligente und optimierte Lösung handelt und nicht um eine digitale Adaption der realen Vorgänge.

4 Technische Grundlagen

Dieses Kapitel beschreibt zum einen die Technologien, die im Projekt angewandt wurden und zum anderen die Technologien, die im Zusammenhang mit der Thematik stehen, aber nicht oder nur teilweise in das Projekt eingeflossen sind, wie z.B. UMTS.

4.1 Modellierung

4.1.1 UML

UML (Unified Modeling Language) ist eine standardisierte Modellierungs-Sprache (bald ISO-Standard). Sie schreibt jedoch keinen bestimmten Softwareentwicklungsprozess vor und kann somit unabhängig von einem Software-Entwicklungsprozess eingesetzt werden.

Die Unified Modeling Language ist eine Sprache zur:

- Spezifikation in verschiedenen Detaillierungsgraden
- Graphischen Darstellung von Softwaresystemen
- Konstruktion und Abbildung von Modellen auf verschiedene Programmiersprachen
- Verwaltung und Erstellung von Dokumentation, die im Rahmen eines Softwareentwicklungsprojekts anfällt
- Erstellung von Modellen für Softwaresysteme, Geschäftsmodelle und andere Nicht-Softwaresysteme

Sie bietet den Entwicklern die Möglichkeit, den Entwurf und die Entwicklung von Softwaremodellen auf einheitlicher Basis zu diskutieren. Die UML wird seit 1998 als Standard angesehen. Sie liegt weiterhin bei der Objekt Management Group (OMG) zur Standardisierung vor.

4.1.2 FunSoft-Netze zur Geschäftsprozessmodellierung

FunSoft-Netze stellen eine Möglichkeit dar, Geschäftsprozesse zu modellieren und diese auf der Basis der Modelle zu analysieren und zu verbessern. Durch das Modellieren gewinnt der Anwender dieser Technik einen Überblick über die Vorgänge, die zum Erreichen eines bestimmten Geschäftszwecks notwendig sind. Zusammen mit der Optimierung der Prozesse ist dann der Grundstein gelegt, um ein effizientes Softwaresystem zu entwickeln, dass die an den Prozessen beteiligten Personen möglichst optimal in ihrer täglichen Arbeit unterstützt und somit deren Produktivität erhöht. FunSoft-Netze basieren auf den in der Softwaretechnik schon seit längerem bekannten Petri-Netzen, sind aber um einige Elemente erweitert worden, um speziell den Anforderungen an das Modellieren von Geschäftsprozessen gerecht zu werden. So bietet gerade die Möglichkeit unternehmensübergreifende Prozesse zu modellieren, einen wesentlichen Vorteil gegenüber herkömmlichen Petri-Netzen. Um einen praktikablen und effizienten Einsatz bei der Erfassung von Geschäftsprozessen mit Hilfe von FunSoft-Netzen zu ermöglichen, stehen dem Anwender mehrere Tools zur Verfügung, die den FunSoft-Netz-Ansatz implementieren. Eines dieser Tools ist das Programm LeuSmart, das von der Firma adesso vertrieben wird.

4.2 Java-Technologie

4.2.1 Java und geeignete Entwicklungsumgebungen

Java wurde von Sun Microsystems 1991 im Rahmen eines Forschungsprojektes entwickelt. Es sollte Software für Verbraucherelektronikgeräte wie Fernseher, Videorekorder oder Toaster entwickelt werden. Ziel war es, eine Programmiersprache zu schaffen, die schnell, effizient und leicht auf vielfältige Hardwareplattformen zu portieren ist.

Bei jedem Softwareprojekt sollte man sich Gedanken machen, welche Programmiersprache man einsetzen will. Die Sprache muss den Anforderungen des Projektes genügen und gleichzeitig auf möglichst vielen Plattformen ausführbar sein. Folgende Eigenschaften sprechen für den Einsatz von Java:

Java ist plattformunabhängig

Die Plattformunabhängigkeit von Java ist der wichtigste Vorteil gegenüber allen anderen Sprachen. Java ist sowohl auf der Quell- als auch der Binärebene plattformunabhängig.

Java ist objektorientiert

Objektorientiertes Programmieren hat sich als eine der wichtigsten Programmiertechniken durchgesetzt. Die Verwendung einer echten objektorientierten Sprache wie Java bietet den Vorteil modularer Programmierung und die Möglichkeit diesen Code auch wiederzuverwenden.

Java ist leicht zu erlernen

Bei der Entwicklung von Java wurden neben Portierbarkeit und Objektorientierung noch weitere Ziele verfolgt. Die Sprache sollte robust und leicht zu erlernen sein. Robustheit bedeutet hierbei, dass das Risiko beim Programmieren schwer abzugrenzende Fehler zu produzieren (z.B. der beliebte „segmentation fault“ beim falschen Umgang mit Pointern in C/C++) gering ist. Trotz dieser Kompaktheit weist Java eine Menge Leistung und Flexibilität auf.

Die meisten der heutigen Entwicklungsumgebungen erfüllen die gleichen Grundvoraussetzungen (Editor, Debugger, Compiler sind vorhanden). Oft werden mehrere Programmiersprachen unterstützt. Daher kann man bei der Auswahl nicht mehr viel falsch machen. Folgende Alternativen standen zur Auswahl:

Umgebung selber zusammenstellen

Die erste Möglichkeit sich eine Entwicklungsumgebung für Java zu schaffen, ist die Zusammenstellung selbst gewählter Werkzeuge. Viele erfahrene Entwickler wählen diesen Weg, weil sie an Werkzeuge aus alten Projekten gewöhnt sind und sich nicht umstellen wollen. Zudem gab es zu Anfang für Java keine eigenen Entwicklungsumgebungen, sodass der Entwickler auf seine „alten“ Werkzeuge angewiesen war. Für den Anfänger ist es aber ratsamer, direkt eine IDE (Integrated Development Environment) zu benutzen.

KAWA 5.0

Die Entwicklungsumgebung läuft nur unter Windows und ist speziell an die Entwicklung von Java-Programmen angepasst worden. Das Programm wird von der Firma Allaire (http://www.allaire.com) in zwei Versionen angeboten: KAWA Professional Edition (mit Editor, Debugger, Plugins) sowie KAWA Enterprise Edition (mit Editor, Debugger, Plugins, EJB-Wizard, Integration eines Application Servers).

JBuilder5 Personal Edition

Die Personal Edition kann unentgeltlich von der Homepage heruntergeladen werden. Diese Version hat nahezu volle Funktionalität und kann zeitlich unbegrenzt genutzt werden. Allerdings sind in dieser Version nicht alle Features des JBuilders enthalten. Die beiden Vollversionen JBuilder5 Professional und JBuilder5 Enterprise mit allen Features sind jedoch sehr teuer. Mit dem JBuilder können Java2 Anwendungen

entwickelt werden. Dabei kommen auch visuelle Werkzeuge zum Einsatz. Das Programm läuft unter Windows, Linux und Solaris.

Together 5.5

Die Wahl fiel auf diese Entwicklungsumgebung. Die Gründe dafür waren in erster Linie die gute Presse, Erfahrungen anderer PGs, die Designwerkzeuge für Projektentwürfe und die vorhandenen Erfahrungen einiger PG-Teilnehmer. Eine detaillierte Beschreibung folgt in den weiteren Kapiteln.

4.2.2 Komponentenbasierte Softwareentwicklung/EJBs

Von der komponentenbasierten Softwareentwicklung verspricht man sich, die drei im Konflikt stehenden Ziele der Softwareentwicklung - Termintreue, Kostentreue und Qualitätstreue - besser beherrschen zu können. Die Schlüssel zum Erfolg sollen dabei in der Wiederverwendung, der Verteilung und der Anwendungsnähe von Komponenten liegen. Enterprise JavaBeans [Sun02] sind auf Java basierende, so genannte verteilte serverseitige Komponenten, deren Spezifikation aus dem Hause Sun Microsystems stammt. Die Fokussierung des Komponentenmodells der Enterprise JavaBeans richtet sich klar auf die Realisierung von verteilten, geschäftskritischen Anwendungen. Für solche Anwendungen sind die Anforderungen an Skalierbarkeit sowie Transaktions- und Sicherheitsdiensten besonders hoch. Enterprise JavaBeans erfüllen diese Anforderungen und bieten darüber hinaus Plattformunabhängigkeit sowie Dienste zu Persistenz, Nebenläufigkeit, Speicherverwaltung und Fehlerbehandlung. Die verschiedenen zur Verfügung stehenden Dienste ermöglichen, dass man sich bei der Entwicklung im Wesentlichen auf die Geschäftslogik konzentrieren kann.

4.2.3 Internettechnologien: Servlets, JavaScript, JSPs

JavaScript ist eine Sprache zur clientseitigen Programmierung. Trotz des Namens und gewisser Ähnlichkeiten, z.B. gleiche Syntax für Schleifen, unterscheidet sich JavaScript von Java deutlich. Genaugenommen sind die Übereinstimmungen so gering, dass der Name „JavaScript“ sehr irreführend ist. JavaScript ist eine Scriptsprache, die nur für Webseiten verwendet wird, wohingegen Java eine universelle Programmiersprache ist. JavaScript ist imperativ, im begrenzten Maße objektorientiert und wird als Quelltext zur Laufzeit interpretiert.

Servlets [servlet] und JSPs sind neue Ansätze zur Programmierung auf der Server-Seite. Servlets und JSPs sind Teil der J2EE (Java2Platform Enterprise Edition) und somit für den industriellen Einsatz in Verbindung mit anderen J2EE Technologien wie EJB (Enterprise JavaBeans) gedacht. Servlets sind also nichts anderes als Java Programme, die die Eigenschaften einer bestimmten API, der Servlet API, ausnutzen. Der Vorteil von Servlets gegenüber beispielsweise herkömmlichen CGI Skripten liegt darin, dass sie nicht der Beschränkung auf eine Plattform unterliegen und somit eine größere Flexibilität erlauben. JSP (Java Server Pages) ist eine auf Servlets aufbauende Technik. In einer JSP-Seite ist das Verhältnis von Java-Code und HTML-Code umgekehrt, der Java-Code ist in HTML eingebettet.

Die Kombination dieser beiden Techniken ermöglicht es, komplexe Web-Anwendungen auf Java Basis zu entwickeln, die eine Arbeitsteilung von Web-Designer und Anwendungs-Entwickler ermöglichen.

4.2.4 Application Server-Komponententechnologie für Enterprise-Computing

IT-Entscheider und Anwendungsentwickler stehen unter ständig zunehmendem Zeit- und Kostendruck. Um die Kosten- und Wettbewerbssituation des Unternehmens positiv zu

beeinflussen, müssen Anwendungen immer schneller den sich ändernden Einsatzanforderungen angepasst werden. Zusätzlich ist mit der Internet- und Web-Technologie eine wesentliche, auch qualitative Änderung der Situation einer Unternehmens-IT verbunden. Der damit mögliche Thin-Client-Einsatz spart zwar erhebliche Kosten für Administration und Hardware, erfordert aber eine leistungsfähige Mehrschicht-Architektur (3-tier-Architektur) des Gesamtsystems, das außerdem die Integration der neuen Anwendungen mit den existierenden Legacy-Systemen leisten muss. Um trotz dieser steigenden Systemkomplexität die Reaktionsschnelligkeit der IT-Entwicklung zu steigern, sind flexible und effiziente Entwicklungstechniken und Tools erforderlich, die nicht nur die Entwicklung der Client-GUIs, sondern auch die Implementierung der zentralen OLTP-Services (Online Transaction Processing) einer 3-tier-Architektur mit gleicher Effizienz ermöglichen. Eigenentwicklungen sollen dabei mit Standard-Komponenten, die am Markt fertig angeboten werden und nur noch konfiguriert werden müssen, ermöglicht werden. Die Spezifikation eines Client/Server-Komponentenmodells, das diesen Ansprüchen gerecht zu werden verspricht, liegt seit April '98 in einer 1. Version vor. Unter dem Namen „Enterprise JavaBeans“ (EJB) hat Sun Microsystems in Zusammenarbeit mit OLTP-Herstellern ein Java-basiertes Server-Komponentenmodell erarbeitet, das als zukünftiges Komponentenmodell für CORBA gehandelt wird. In Kombination mit der bewährten TP-Monitor Funktionalität als leistungsfähige Application Server-Infrastruktur für eine sichere und robuste verteilte Ablaufumgebung, entsteht damit ein Applikations-Framework, das Effizienz, Schnelligkeit, Flexibilität, Skalierbarkeit, Zuverlässigkeit und Sicherheit für die IT-Abteilungen und die Benutzer Realität werden lassen. Kurz: Es geht bei Application Servern um eine Laufzeitumgebung für verteilte Anwendungen, die mit Java kooperiert und durch das EJB Komponentenmodell realisiert wird.

4.3 Mobiltechnik

4.3.1 Usability Engineering für mobile Endgeräte

Ein wesentlicher Punkt beim Usability Engineering ist der Entwurf einer geeigneten Benutzerschnittstelle. Im Zusammenhang mit mobilen Endgeräten sind dabei einige Besonderheiten zu beachten. So sind einerseits die Möglichkeiten zur Datenein- und -ausgabe sehr begrenzt, andererseits ist auch die Art der Anwendung nicht mit der Benutzung eines Webbrowsers zu vergleichen. Schließlich dürfen technische Aspekte wie hohe Latenzzeiten oder geringe Bandbreite nicht vergessen werden.

Eine Studie über WAP-Mobiltelefone zeigt den Stand der Technik bei mobilen Anwendungen. Hier wird besonders deutlich, dass die Fehler bei der Umstellung von herkömmlichen Medien auf Internetpräsentationen wiederholt wurden und daher in diesem Bereich noch deutliches Verbesserungspotential erkennbar ist.

4.3.2 WAP – Das mobile Netz

WAP steht für „Wireless Application Protocol“, eine Spezifikation, deren Entwicklung in der Industrie stattgefunden hat, um multimediale Inhalte auch auf mobile Geräte, z.B. Handys, zu bringen. Ein Teil dieser Spezifikation ist die Wireless Markup Language (WML), die das entsprechende Gegenstück zu HTML darstellt.

4.3.3 UMTS

Mit dem „Universal Mobile Telecommunications System“ (UMTS) wird ein Mobilfunkstandard der 3. Generation eingeführt, der breitbandige Datendienste mobilen Teilnehmern verfügbar machen soll. Obwohl dieses Thema in der Öffentlichkeit breit diskutiert wird, sind die Bedingungen für den Einsatz dieser Technologie zum jetzigen Zeitpunkt zu schlecht. Die extrem schlechte Verfügbarkeit von Testgeräten und das nur für Testzwecke ausgelegte Netz machen eine Entwicklung für diesen Standard unmöglich.

4.4 Versionsmanagement

Ein Source-Code-Control-System ist eine Sammlung von Tools, um Modifikationen an Quelltextdateien zu kontrollieren, zu verfolgen und zu dokumentieren. Das Verfolgen der Modifikationen erfolgt in der Regel so, dass zu jedem Zeitpunkt vorherige Versionen der betrachteten Datei wiederhergestellt werden können. Diese Tools werden in Verbindung mit Programmquelltexten genutzt, können aber problemlos für andere textbasierte Dateitypen wie z.B. Textdateien oder Dokumentationstexte eingesetzt werden. Das erste bekannte Versionsmanagement-System, dessen Grundidee sich bis heute nicht geändert hat, wurde bereits auf frühen Unix-Systemen entwickelt und eingesetzt. Innerhalb des Seminars in der Vorbereitungsphase der PG wurden die Systeme SCCP, RCS, CVS, VSS und ClearCase sowie deren Integration in Java-Entwicklungsumgebungen vorgestellt.

5 Projektmanagement

5.1 Wahl des Vorgehensmodells

Die systematische Vorgehensweise bei der Entwicklung eines Softwaresystems entscheidet im hohen Maße über dessen Erfolg oder Misserfolg. Die einzelnen Aktivitäten eines solchen Prozesses sollten sorgfältig geplant und verfolgt werden. Am Anfang eines jeden Softwareprojektes sollte daher die Frage nach einem geeigneten Vorgehensmodell stehen.

Die Wahl wurde im Wesentlichen durch die folgenden Faktoren beeinflusst:

1. Vorgaben durch die PG Betreuung – d.h. Meilensteine, wie etwa die Fertigstellungstermine für Geschäftsprozessdokumentation und Entwurf, waren vorgegeben und zu berücksichtigen.
2. Mangelndes Wissen über Alternativen auf Seiten der PG Teilnehmer – dieser Umstand ist nicht zuletzt auf die Situation an der Universität Dortmund zurückzuführen, da dort derzeit keine (Stamm-)Vorlesung angeboten wird, die die Thematik „Vorgehensmodelle“ angemessen behandelt.

Unter der Berücksichtigung dieser beiden Faktoren fiel unsere Wahl auf das Wasserfallmodell, welches im Folgenden kurz erläutert werden soll.

5.2 Das Wasserfallmodell

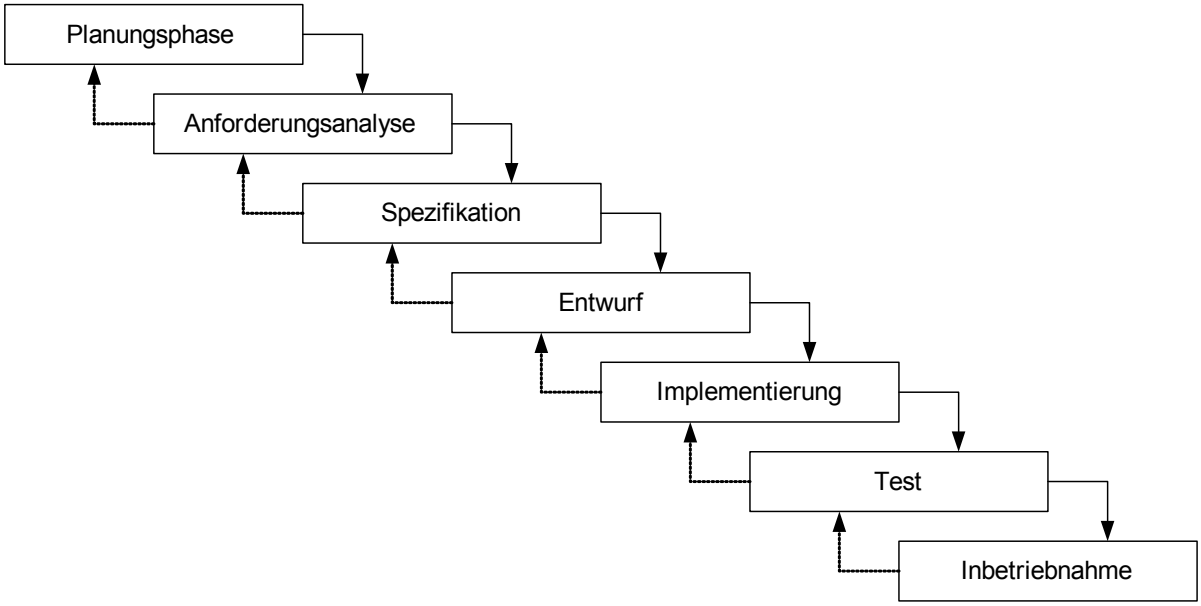


Abbildung 5-1: Das Wasserfallmodell

Beim Wasserfallmodell werden die verschiedenen Phasen des Entwicklungsprozesses sequenziell hintereinander durchgeführt, d.h. beim Übergang von einer Phase zur nächsten wird vorausgesetzt, dass die vorhergehende Phase abgeschlossen wurde. Nur in Fehlerfällen sind Rückschritte zur vorhergehenden Phase möglich. Am Ende einer jeden Phase steht ein Dokument, welches im Projektplan als Meilenstein geplant wird, und das als Grundlage für die folgende Phase dient. Das Wasserfallmodell zeichnet sich also weniger durch die in Abbildung 5-1 abgebildeten konkreten Phasen ab, als vielmehr durch seine namensgebende Struktur.

5.3 Projektplan und tatsächlicher Verlauf

Nr.	Vorgangsname	Dauer	Anfang	Ende
1	**Projekt "Nighshift"**	**261 Tage**	**Mo 15.10.01**	**Fr 11.10.02**
2	**PG Sitzung Wintersemester**	**119 Tage**	**Mo 15.10.01**	**Do 28.03.02**
51	**PG Sitzung Sommersemester**	**137 Tage**	**Mo 01.04.02**	**Mo 07.10.02**
80	**Entwicklung**	**246 Tage**	**Mo 15.10.01**	**Fr 20.09.02**
81	**Planungsphase**	**29 Tage**	**Mo 15.10.01**	**Do 22.11.01**
82	Projektplan erstellen	6 Tage	Mo 15.10.01	Mo 22.10.01
83	Informationsermittlung Ist-Zustand (Recherche, Interviews, ...)	9 Tage	Mo 15.10.01	Do 25.10.01
84	Derzeitige Geschäftsprozesse modellieren und dokumentieren	10 Tage	Fr 26.10.01	Do 08.11.01
85	Informationsermittlung Optimierun (Recherche, Interviews, ...)	9 Tage	Mo 22.10.01	Do 01.11.01
86	Optimierte Geschäftsprozesse modellieren und dokumentieren	10 Tage	Mo 05.11.01	Fr 16.11.01
87	Review Dokumentation der Geschäftsprozesse	4 Tage	Mo 19.11.01	Do 22.11.01
88	Dokumentation der Geschäftsprozesse erstellt	0 Tage	Do 22.11.01	Do 22.11.01
89	**Anforderungsanalyse**	**17 Tage**	**Do 22.11.01**	**Fr 14.12.01**
90	Anforderungsdokument erstellen	13 Tage	Do 22.11.01	Mo 10.12.01
91	Review Anforderungsdokument	4 Tage	Di 11.12.01	Fr 14.12.01
92	Anforderungsdokument erstellt	0 Tage	Fr 14.12.01	Fr 14.12.01
93	**Spezifikation**	**17 Tage**	**Mi 28.11.01**	**Do 20.12.01**
94	Spezifikationsdokument erstellen	13 Tage	Mi 28.11.01	Fr 14.12.01
95	Review Spezifikationsdokument	4 Tage	Mo 17.12.01	Do 20.12.01
96	Spezifikationsdokument erstellt	0 Tage	Do 20.12.01	Do 20.12.01
97	**Entwurfsphase**	**34 Tage**	**Mo 07.01.02**	**Do 21.02.02**
98	**Einführung und Planung der Entwurfsphase**	**4 Tage**	**Mo 07.01.02**	**Do 10.01.02**
99	Präsentation einiger Beispiele EJB bzw. JSP Umfeld	1 Tag	Mo 07.01.02	Mo 07.01.02
100	Weitere Planung der Entwurfsphase	3 Tage	Di 08.01.02	Do 10.01.02
101	**Entwurfsdokument erstellen**	**28 Tage**	**Fr 11.01.02**	**Di 19.02.02**
102	Grobentwurf	11 Tage	Fr 11.01.02	Fr 25.01.02
103	Feinentwurf	17 Tage	Mo 28.01.02	Di 19.02.02
104	Storyboard erstellen	22 Tage	Do 17.01.02	Fr 15.02.02
105	Review Storyboard	2 Tage	Mo 18.02.02	Di 19.02.02
106	Review Entwurfsdokument	2 Tage	Mi 20.02.02	Do 21.02.02
107	Entwurfsdokument erstellt	0 Tage	Do 21.02.02	Do 21.02.02

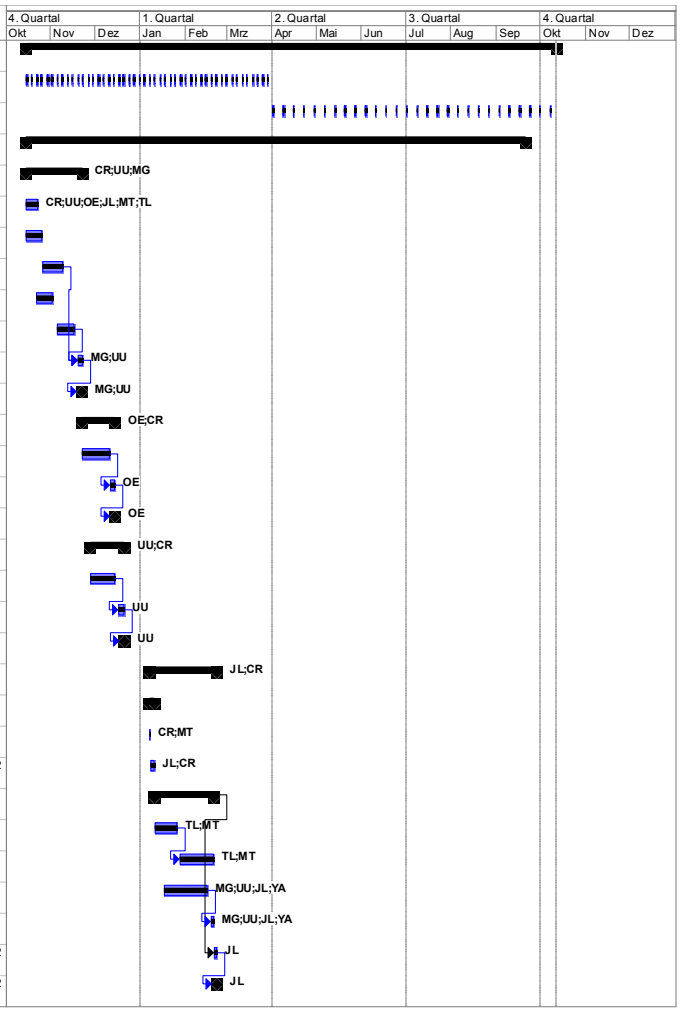


Abbildung 5-2: Der Projektplan (Teil 1)

Nr.	Vorgangsname	Dauer	Anfang	Ende
108	**Implementierungsphase**	**148 Tage**	**Do 28.02.02**	**Fr 20.09.02**
109	Geschäftslogik	148 Tage	Do 28.02.02	Fr 20.09.02
110	Benutzeroberfläche	148 Tage	Do 28.02.02	Fr 20.09.02
111	**Zwischenbericht**	**39 Tage**	**Mo 07.01.02**	**Do 28.02.02**
112	Zwischenbericht erstellen	33 Tage	Mo 07.01.02	Mi 20.02.02
113	Review Zwischenbericht	6 Tage	Do 21.02.02	Do 28.02.02
114	Zwischenbericht erstellt	0 Tage	Do 28.02.02	Do 28.02.02
115	**Endbericht**	**50 Tage**	**Mo 05.08.02**	**Fr 11.10.02**
116	Endbericht erstellen	40 Tage	Mo 05.08.02	Fr 27.09.02
117	Review Endbericht	10 Tage	Mo 30.09.02	Fr 11.10.02
118	Endbericht erstellt	0 Tage	Fr 11.10.02	Fr 11.10.02
119	**Qualitätsmanagement**	**245 Tage**	**Mo 15.10.01**	**Do 19.09.02**
120	Konzept Versionskontrolle (CVS)	6 Tage	Mo 22.10.01	Mo 29.10.01
121	Dokumentvorlage Protokolle	0 Tage	Mo 15.10.01	Mo 15.10.01
122	Dokumentvorlage Anforderungsdokum	0 Tage	Do 22.11.01	Do 22.11.01
123	Dokumentvorlage Spezifikationsdokum	0 Tage	Mi 28.11.01	Mi 28.11.01
124	Dokumentvorlage Zwischenbericht	0 Tage	Mo 07.01.02	Mo 07.01.02
125	Dokumentvorlage Entwurfsdokument	0 Tage	Mo 07.01.02	Mo 07.01.02
126	Dokumentvorlage Endbericht	0 Tage	Fr 14.06.02	Fr 14.06.02
127	**Zyklischer Review Projektplan**	**222 Tage**	**Do 15.11.01**	**Do 19.09.02**
139	**Öffentlichkeitsarbeit**	**241 Tage**	**Mo 22.10.01**	**Mo 23.09.02**
140	Entwurf & Realisierung Website	3 Tage	Mo 22.10.01	Mi 24.10.01
141	Website erreichbar	0 Tage	Mi 31.10.01	Mi 31.10.01
142	Veröffentlichung des Projekts im Unireport	0 Tage	Mo 08.04.02	Mo 08.04.02
143	Projektpräsentation auf dem Studieninformationstag (Campusfest)	0 Tage	Sa 06.07.02	Sa 06.07.02
144	Präsentation der Ergebnisse (Abschlusspräsentation)	0 Tage	Mo 23.09.02	Mo 23.09.02
145	**Infrastruktur (HW & SW)**	**20 Tage**	**Do 18.10.01**	**Do 15.11.01**
146	HW & SW bestellt	0 Tage	Do 18.10.01	Do 18.10.01
147	HW & SW geliefert	0 Tage	Mi 07.11.01	Mi 07.11.01
148	Hardwareaufbau	5 Tage	Mi 07.11.01	Di 13.11.01
149	Softwareinstallation	5 Tage	Mi 07.11.01	Di 13.11.01
150	Infrastruktur betriebsbereit	0 Tage	Do 15.11.01	Do 15.11.01

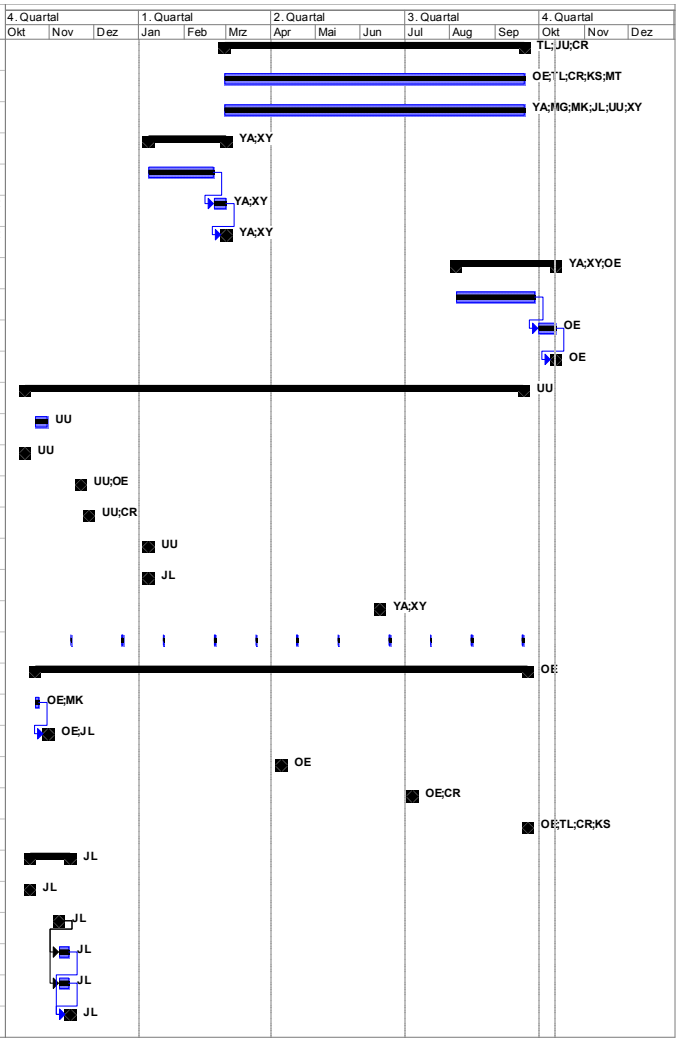


Abbildung 5-3: Der Projektplan (Teil 2)

Der in Abbildung 5-2 bzw. Abbildung 5-3 dargestellte Projektplan entspricht dem der letztmaligen Revision vom 23.09.2002. Er zeigt den jeweiligen Vorgang, seine Dauer und sein Anfangs- bzw. Enddatum. Die im Balkendiagramm wiedergegebene Ressourcenzuordnung ist als reine Zuordnung der Verantwortlichkeit zu verstehen und wurde nicht zur Ressourcenplanung im herkömmlichen Sinne (→Auslastung, Kosten, …) verwendet.

5.3.1 Planungsphase

In der Planungsphase wurde zunächst ein Projektplan in einer ersten initialen Version erstellt. Neben dem eigentlichen Entwicklungsprozess versuchten wir dabei direkt Qualitätsmanagements-, Öffentlichkeitsarbeits- und Berichtsaspekte zu planen und entsprechend im Projektplan festzuhalten.
Danach sammelten wir Information rund um die Geschäftsprozesse im Fachbereich Informatik (FBI). Mit den gewonnen Informationen modellierten und dokumentierten wir zunächst den Ist-Zustand am Fachbereich und erarbeiteten darauf aufbauend mögliche Optimierungen, die wir ebenfalls modellierten und dokumentierten. Die Gruppe wurde dazu in zwei Untergruppen aufgeteilt, d.h. eine befasste sich mit dem Bereich Übungs-, die andere mit dem Prüfungsmanagement.

Probleme in dieser Phase

Gleich zu Beginn unseres Projektes kam es zu erheblichen Verzögerungen, da uns die benötigten Arbeitsrechner nicht rechtzeitig zur Verfügung gestellt werden konnten. Um unser Ziel, das Anforderungs- und das Spezifikationsdokument noch vor Jahreswende fertig zu

stellen, erreichen zu können, waren wir gezwungen die beiden folgenden Phasen weitgehend zu parallelisieren.

5.3.2 Anforderungsanalyse

In der Anforderungsanalyse sammelten wir mögliche Anforderungen an unser System, klassifizierten diese in „kann“ und „muss“ und schafften uns so eine dokumentierte Basis gegen die wir später implementieren konnten.

Probleme in dieser Phase
Es bereitete einem Teil der Gruppe Probleme die Anforderungen exakt zu erfassen und zu formulieren. Inwieweit dies auf mangelnde Erfahrung zurückzuführen ist sei einmal dahingestellt. Aus heutiger Sicht hätte das Anforderungsdokument wohl nicht so kommentarlos angenommen werden dürfen wie dies letztlich geschehen ist.

5.3.3 Spezifikation

Aufbauend auf den gesammelten Anforderungen bestand der nächste Schritt in der Spezifikation unseres Systems. Dazu beschrieben wir das Verhalten unseres Systems aus Anwendersicht und veranschaulichten dieses anhand von Modellen und Diagrammen.

Probleme in dieser Phase
Die Probleme, die während der Spezifikation auftraten, lassen sich mit denen der Anforderungsanalyse vergleichen. Es sei daher an die entsprechende Stelle verwiesen.

5.3.4 Entwurf

Basierend auf den Anforderungen und der Spezifikation machten wir uns nun an das softwaretechnische Design unseres Systems. Dabei hielten wir uns an die mehrschichtige Architektur der Java Enterprise Edition (J2EE). Neben der fachlichen Unterteilung der Gruppe (siehe oben) unterteilten wir nun zusätzlich nach eher technischen Belangen, also in eine Gruppe zuständig für die „Geschäftslogik“ und eine für die „Benutzeroberfläche“.

Probleme in dieser Phase
Das größte Problem dieser Phase war die mangelnde Erfahrung mit der gewählten Architektur. Ein großer Teil des Entwurfs wurde daher auch gleich zu Anfang der Implementierung verworfen. Insgesamt wurde die Arbeit dieser Phase zu wenig dokumentiert und es wurde nicht genug darauf geachtet, den Entwurf jedem Gruppenmitglied „nahe zu bringen“.
Bis zu diesem Zeitpunkt waren wir weitgehend in unserem zuvor festgesetzten Plan geblieben. Die bis hierher abgeschlossenen Phasen ließen sich allerdings auch mit den vorhandenen Ergebnissen einfach als beendet erklären.

5.3.5 Implementierungsphase

Wir entschlossen uns vom Wasserfallmodell abzuweichen und von nun an in kurzen Iterationsphasen von Implementierung, Test und Integration fortzufahren. Die wöchentlichen PG Sitzungen dienten uns dazu die Entwicklungsarbeiten zu koordinieren und etwaige Probleme zu besprechen.

Probleme in dieser Phase
Das Ende der Implementierungsphase verzögerte sich um insgesamt 2,5 Monate. Immer wieder gab es technische Probleme, die die Arbeit zeitweise vollständig zum Erliegen brachten. Auch hier machte sich wieder die mangelnde Erfahrung mit der Architektur bemerkbar.
Gerade vor den „harten Terminen“ (Studieninformationstag bzw. Endpräsentation) erwies sich zudem die mangelnde Ressourcenausstattung in produktiver Hinsicht als Hemmschuh.

5.3.6 Erkenntnisse und Ergebnisse

Die Zeitverzögerung des Projekts ist nicht zuletzt auf die

- unzureichende Ressourcenausstattung
- fehlende Motivation
- dürftige Einsatzbereitschaft
- unzulängliche Betreuung
- ungenügende Erfahrung
- sowie mangelnde Teamfähigkeit

zurückzuführen.

Sinn einer Projektgruppe ist es sicherlich, sich mit den oben genannten Problematiken auseinander zusetzen und an deren Bewältigung zu lernen. Damit aus einer Projektgruppe aber nicht ein reines „so sollte man es nicht machen“ wird, sollte man vielleicht einen Teil der Problematiken „präventiv“ um- bzw. angehen.

Was könnte man also besser machen?

- Die Projektgruppe ist eine Lehrveranstaltung, dies sollte angemessene Berücksichtigung finden. Insbesondere Vorgehen, beteiligte Rollen und etwaige Ergebnisdokumente eines Projektes sollten ausreichend thematisiert werden. Auf immer wiederkehrende Problematiken sollte zudem bereits in der Einführungsphase (Einführungsseminar) eingegangen werden.
- Problematisch am Vorgehen zeigte sich, dass die gewonnenen Erkenntnisse am Ende einer Phase nicht (erneut) zur Anwendung gebracht werden konnten. Vielleicht sollte man sich daher im Rahmen einer Projektgruppe über alternative Vorgehensmodelle Gedanken machen, die einen iterativen Ansatz haben.
- Es zeigte sich immer wieder, dass kein einheitliches Verständnis darüber herrschte, wie viel Zeit von jedem Projektteilnehmer in die Projektarbeit investiert wird. Man sollte daher frühzeitig Sorge dafür tragen, dass diese Frage geklärt wird.
- Die unzureichende und verspätete Ressourcenausstattung (Raumangebot viel zu knapp für 11 Teilnehmer, Hardware für nur 4 Arbeitsplätze) erwies sich mehr als einmal als verschleppend und damit demotivierend. Hier sollte man darauf achten, dass bereits im Vorfeld eine entsprechende Ausstattung vorgesehen wird.

6 Qualitätsmanagement

Zur Sicherstellung der Qualität unseres Softwareprojekts wurde zu Beginn ein Qualitätsmanagementplan erstellt. Durch das Qualitätsmanagement können frühzeitig Fehler in Konzeption und Entwicklung erkannt und direkt vermieden werden. Durch regelmäßige Durchführung von Qualitätssicherungsmaßnahmen erhält man eine höhere Transparenz des Projektstatus und damit eine Erleichterung der Projektsteuerung. Das Qualitätsmanagement gliedert sich in die drei Bereiche:

1. Qualitätsplanung
2. Qualitätssicherung/Qualitätsprüfung
3. Ziele und Maßnahmen.

6.1 Qualitätsplanung

In dieser Phase wurden durch das Projektteam Qualitätsziele bzw. Qualitätskriterien festgelegt. Beispiele für Qualitätsziele sind Performanz, Ergonomie, Termintreue, Wartbarkeit, etc. Unter Qualitätskriterien fallen zum Beispiel „Jeder Dialogwechsel muss innerhalb von 2 Sekunden geschehen“. Durch die Qualitätskriterien kann das Erreichen der Qualitätsziele gemessen werden. Nach Festlegung der Ziele einigte sich das Projektteam auf Qualitätsmaßnahmen, durch die die Qualitätsziele erreicht werden sollen

6.2 Qualitätssicherung/Qualitätsprüfung

Die Qualitätssicherung dient zur Bestätigung des Erreichens eines Qualitätsziels und zur Erkennung von Fehlern und Mängeln. Dazu werden analytische Qualitätssicherungs-maßnahmen in allen Projektphasen durchgeführt. Beispiele solcher Maßnahmen sind Dokumenten-Review, Code-Review, Komponententest und Integrationstest.

6.3 Ziele und Maßnahme

6.3.1 Qualitätsziele für die PG nightshift

Die Projektgruppe hat sich in einer Diskussion auf folgende Qualitätsziele geeinigt (mit angegebener Priorität):

1. Umsetzung der Grundfunktionalität (Kriterium: Anforderungsdokument)
2. Einhaltung des Abgabetermins (Termin: Ende des 2. PG-Semesters)
3. Erreichen einer Gesamtqualität mit folgenden Unterprioritäten:
    - Sicherheit
    - Benutzbarkeit
    - Stabilität
    - Skalierbarkeit
4. Spaß bei der Arbeit

6.3.2 Qualitätsmaßnahmen

Die Projektgruppe einigte sich auf die folgenden Maßnahmen zur Erreichung der festgelegten Qualitätsziele. Diese wurden unterteilt in organisatorische und technische Maßnahmen.

6.3.2.1 Organisatorische Maßnahmen

Regelmäßige Dokumentenreviews

Alle Dokumente, die im Laufe des Projekts immer wieder angepasst wurden, wurden vom Qualitätsmanager und dem Verantwortlichen für das Dokument regelmäßig überprüft. Die Rolle des Qualitätsmanagers kam dabei nicht so sehr zu tragen. Fast immer lasen mehrere oder auch alle PG-Teilnehmer die Dokumente gegen und unterstützten so den Verantwortlichen für das jeweilige Dokument. Die Termine dazu wurden im Projektplan eingetragen.

Strukturierte Dokumentenablage

Alle Dokumente wurden an einem zentral für alle zugänglichen Ort abgelegt. In der PG nightshift wurde dazu ein CVS-Server aufgesetzt und wie folgt strukturiert:

- doc (Dokumentenablage)
  - interview (Interviews mit Sachverständigen)
  - links (URLs zu Dokumenten im Internet)
  - modellierung (Alle Dokumente, die zur Modellierung des Systems gehören (z.B. LeuSmart-Diagramme))
  - organisatorisch (Alle Dokumente, die zur Organisation des Projekts dienen (z.B. Projektplan, Anforderungsdokument usw.))
  - protokolle (Alle Sitzungsprotokolle)
  - technisch (Alle Dokumente, die mit der Umsetzung des Projekts zu tun haben (z.B. externe APIs, eigene Dokumentation, usw.)
  - veroeffentlichungen (Artikel im UniReport)
  - vortraege (Alle Vorträge in der PG)
- lib (Verwendete Java Bibliotheken)
- script (SQL Skripte für die UPDB)
- src (alle Quelltexte)

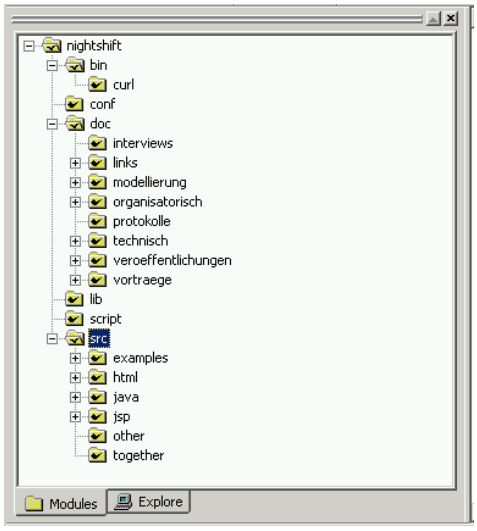


Abbildung 6-1 Struktur im CVS

Checklisten für regelmäßige Aufgaben und Abnahmen

Fielen während der PG regelmäßige Aufgaben an, wurden für deren Abwicklung Checklisten angelegt und bei Durchführung abgearbeitet. Auch Abnahmen wurden über zuvor angelegte Checklisten abgearbeitet.

Testfallplanung

Für die Komponententests und den Integrationstest wurden Testpläne angelegt. Diese wurden größtenteils automatisch abgearbeitet.

6.3.2.2 Technische Maßnahmen

Programmierrichtlinien

In den Programmierrichtlinien wurde festgelegt, wie der Programmcode auszusehen hat. Auch die Namenkonventionen wurden hier festgelegt. Kommentare im Quellcode wurden automatisch von Together erstellt und vom Entwickler ausgefüllt. Das Dokument wurde vom Qualitätsmanager erstellt.

Regelmäßige Code-Reviews

In regelmäßigen Abständen wurde der Programmcode auf die Programmierrichtlinien hin überprüft.

Erst Testen

Dieses Vorgehen ist dem Extreme Programming entnommen. Dabei wurde vor der Implementierung einer Komponente erst ein Test geschrieben. Diese Testklasse erbte von einer Supertestklasse über die die Auswertung abläuft. Somit war der Komponententest voll automatisch abzuarbeiten.

Fortlaufende Integration

Es existierte ein Integrationssystem, auf welchem nicht entwickelt wurde. Auf diesem System wurde immer die aktuellste, lauffähige Version des Systems gehalten. Neue Funktionalität wurde hier eingespielt und getestet. Verlief der Test positiv, verblieben die neuen Komponenten auf dem System, ansonsten wurden sie wieder entfernt und auf den Entwicklungssystemen weiterentwickelt. Somit war immer ein lauffähiges System vorhanden.

Programmieren in Paaren

Auch dieses Vorgehen stammte aus dem Extreme Programming. Es diente dazu alle Entwickler schnell auf einen Stand zu bringen. Hierbei arbeiteten die Entwickler in Zweierteams zusammen an einem Rechner. Der Teampartner könnte dabei immer frei ausgewählt werden.

Kurze Iterationen und Iterationsplanung

Es wurde in kurzen Iterationsphasen implementiert. Das heißt am Anfang der Phase wurde festgelegt, welche Funktionalität von welchem Entwickler implementiert werden soll. War die Umsetzung abgeschlossen und die neuen Komponenten auf dem Integrationssystem eingespielt, erfolgte ein kritischer Rückblick auf die Iteration.

Zwischenabnahmen

Während der Implementierungsphase wurden Akzeptanztests vom Qualitätsmanager durchgeführt

7 Systementwurf

7.1 Ist-Analyse

7.1.1 Einleitung

In diesem Kapitel wird der Ist-Zustand der Geschäftsabläufe für die Prüfungsanmeldung und das Übungsgruppenmanagement im Fachbereich Informatik (FBI) der Universität Dortmund beschrieben.

Dieses Dokument behandelt die Geschäftsprozesse im Fachbereich Informatik an der Universität Dortmund. Zu Anfang erfolgt ein kurzer Abriss über die Struktur des Dokumentes und es werden die in den Ablaufdiagrammen benutzten Symbole erläutert.

In diesem Kapitel werden zunächst die Ist-Geschäftsprozesse zu Beginn des Wintersemesters 01/02 dargestellt. Dabei sei angemerkt, dass diese auch nur einen exemplarischen Charakter haben, da sich bei der Analyse der Geschäftsprozesse schnell zeigte, dass viele Abläufe nicht eindeutig definiert sind und daher oft von den Verantwortlichen improvisiert werden muss.

Die Analyse erfolgte mit Hilfe von FunSoft-Netzen. Zusätzlich wurden die Grafiken durch erläuternde Texte ergänzt. Alle Modelle wurden mit der Software LeuSmart der Firma adesso erstellt. In dieser Tabelle sind die Symbole aus den LeuSmart Diagrammen erläutert, die in den folgenden Kapiteln den Ablauf der Geschäftsprozesse widerspiegeln sollen.

	Informationsspeicher (Startpunkt)
	Informationsspeicher
	Tätigkeit (ohne spätere Verfeinerung in weiterem Modell)
	Verfeinerung (ein weiteres Modell wurde generiert, in dem der Ablauf genauer beschrieben wird)
	Einstiegs- und Ausstiegspunkt in ein Verfeinerungsmodell
	Tätigkeit, die automatisch (vom Computer) erledigt wird
	Tätigkeit, die manuell ausgeführt wird

Abbildung 7-1: Symbole in Leu Smart Diagrammen

7.1.2 Geschäftsprozessmodellierung mit LeuSmart

Die Applikation LeuSmart von der Firma adesso bietet eine komfortable Oberfläche zur Erstellung und Simulation von Prozessen und Prozesssystemen mit FunSoft-Netzen. Ein LeuSmart-Diagramm ist sehr praxisnah und die anschließende Simulationen kann in den meisten Fällen ohne weitere Interpretation ausgewertet werden. Während der gesamten Modellierungsphase können Elemente bequem mit Hilfe von Kontextmenüs und dem Drag & Drop -Verfahren bearbeitet werden, wobei es auch möglich ist, mehrere Elemente mit Standardverfahren (bekannt aus dem Verhalten des Betriebsystems Windows) zu gruppieren

und so eine effektivere Bearbeitung zu gewährleisten. Die resultierenden Diagramme sind einfach und übersichtlich. Seitens der Projektbetreuer war eine Modellierung mit Hilfe von LeuSmart vorgegeben.

Erfahrungen und Probleme
Kleinere Diagramme ließen sich mit LeuSmart einfach und schnell erstellen. Das Modellieren größerer Diagramme mit Verbindungen untereinander war wesentlich schwieriger, da das hierbei anzuwendende Vorgehen den PG -Mitgliedern nicht ausreichend bekannt war.
Weiterhin erwies es sich als äußerst mühsam, Abbruchbedingungen für zirkuläre Prozesse zu modellieren. In vielen Fällen war nicht offensichtlich, welche Nebenwirkungen ein zusätzlich eingefügtes Token in einem System haben wird. Eine weitere Schwierigkeit ergab sich bei dem Versuch den Initialzustand von komplexeren Prozessen zu modellieren.
Zu den geschilderten Problemen kam die Tatsache, dass sich LeuSmart nur auf Windows 2000 Systemen oder entsprechenden Nachfolgeprodukten der Firma Microsoft installieren ließ. Der Versuch die Software auf einem Windows 98 System zu installieren, führte zu einem nicht lauffähigen System.

7.1.3 Ist-Analyse und Modellierung der Geschäftsprozesse für das Übungsgruppenmanagement

Die Beschreibung und Analyse des Ist-Zustands bezieht sich hauptsächlich auf das Vorgehen von Herrn Dißmann, Lehrstuhl X, da er schon mehrfach Hauptverantwortlicher für die Übungsgruppenveranstaltungen im Fachbereich Informatik gewesen ist und sich sein Abwicklungsmodell bewährt hat. Das Vorgehen an anderen Fachbereichen ist nicht berücksichtigt und könnte komplett anders verlaufen.

Bei der Recherche und Analyse der Geschäftsprozesse stellte sich heraus, dass eigentlich kein klares Konzept vorliegt. Die gesamte Verantwortung der Koordination trägt der Hauptverantwortliche. Er hat komplett freie Hand und schon kleine Fehler seinerseits können dazu führen, dass der gesamte Übungsbetrieb empfindlich gestört wird oder erst gar nicht rechtzeitig beginnen kann.

Die hier vorgestellten Rollen werden auch im Kapitel bzgl. des Übungsgruppenmanagements benutzt. Der Leser sollte diese nicht mit den Rollen des Prüfungsmanagements verwechseln.

Veranstalter	Derjenige, der die Vorlesung (und damit die Übungsgruppe) anbietet
Verantwortlicher	Eine Person, die für alle Übungsveranstaltungen verantwortlich ist
Teilnehmer	Teilnehmer an einer Übungsgruppe, meist Studenten
Übungsgruppenleiter	Die Person, die die Übungsgruppe leitet

Abbildung 7-2: Rollenbeschreibung für das Übungsmanagement

Dieses Modell zeigt den Ablauf einer Übungsgruppenveranstaltung in einer Übersicht. Wichtige Abläufe werden in separaten Modellen näher beschrieben.

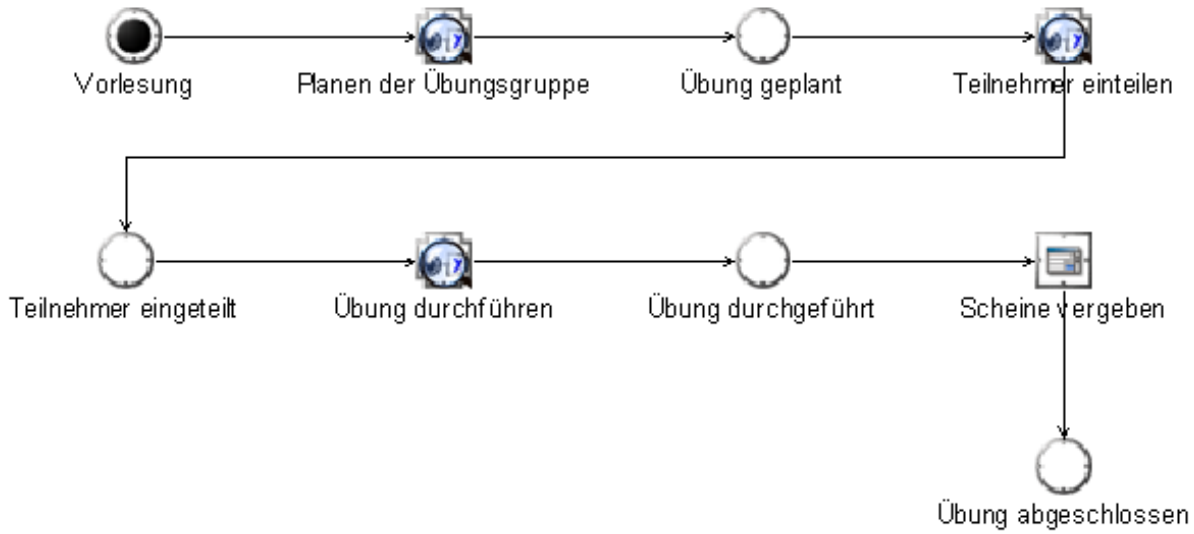


Abbildung 7-3: Ablauf einer Übungsgruppenveranstaltung

Planen der Übungsgruppe
Der Geschäftsprozess des Übungsgruppenmanagement ist sehr linear aufgebaut. Er teilt sich in verschiedene Phasen. Alles beginnt damit, dass ein Professor eine Vorlesung für das kommende Semester plant und dafür auch eine Übung anbieten will. Er meldet diese Vorlesung bei dem Verantwortlichen für alle Übungen an, der dann schließlich die weitere Planung und Organisation übernimmt.

Teilnehmer einteilen
Wenn die Übung von Seiten der Veranstalter und Übungsgruppenleiter organisiert ist, werden die Studenten den einzelnen Übungsgruppen zugeordnet. Dies kann auf verschiedene Arten geschehen und benötigt meist den Zeitraum der ersten Vorlesungswoche, in der mangels vermittelten Vorlesungsstoffes keine Übungsgruppen stattfinden.

Übung durchführen
Als nächstes folgt der eigentliche Übungsbetrieb. Dieser Prozess dauert viele Wochen und gerade hier lohnt sich eine Optimierung des Geschäftsprozesses besonders, da sämtliche Aktivitäten immer wieder in einem Zyklus durchlaufen werden.

Scheine vergeben
Abschließend werden dann die Leistungsnachweise ausgegeben. An dieser Stelle haben sich verschiedene Methoden etabliert. Bei der ersten werden die Leistungsnachweise in den Übungsgruppen ausgegeben. Der Übungsgruppenleiter hat also dafür Sorge zu tragen, dass sie bei der letzten Veranstaltung vorliegen. Bei der zweiten Methode werden die Daten von den Übungsgruppenleitern gesammelt und danach werden alle Leistungsnachweise zentral hinterlegt, z.B. bei der zuständigen Sekretärin. Dort können die Studenten sie abholen. Alternativ kann auch eine Liste mit Teilnehmern direkt an das ZPA geschickt werden, die ihren Übungsschein erhalten haben.

Verfeinerung „Planen der Übungsgruppe“
Dieses Modell beschreibt den genauen Ablauf der Planung einer Übungsgruppen-veranstaltung, wie er zurzeit im Fachbereich Informatik abläuft.

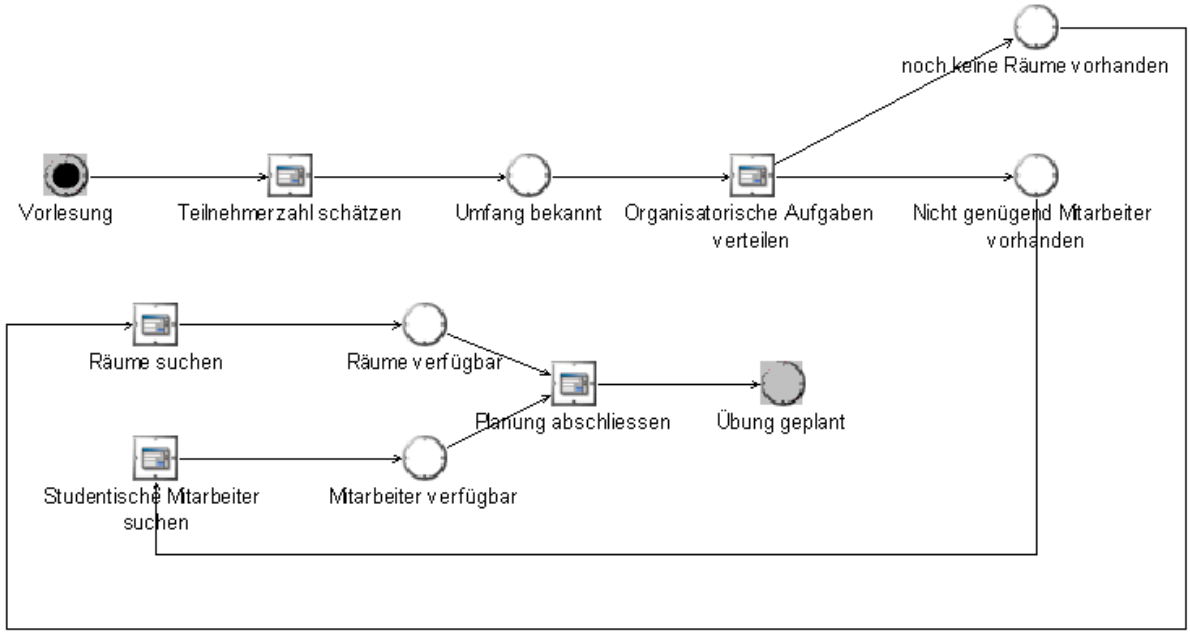


Abbildung 7-4: Ablauf der Planung einer Übungsgruppenveranstaltung

Schätzen der Teilnehmerzahl
Nachdem die Vorlesung von einem Lehrstuhl angekündigt worden ist, schätzt der Hauptverantwortliche die Anzahl der Studenten, die an den Übungen teilnehmen wollen. Wichtige Informationen, die für die Schätzung relevant sind, sind Daten über die Anzahl der Teilnehmer an derselben Veranstaltung in den letzten Semestern. Die Informationen über die Übungsveranstaltung (Anzahl der Teilnehmer, erreichte Punktzahlen, usw.) jedes Semesters werden deswegen gespeichert. Nach dem Schätzen der Teilnehmerzahlen wird der grobe Umfang der Übungsgruppe bestimmt. Danach können die Anzahl der Übungsgruppen, die Größe der Gruppen, die Anzahl der dazu gebrauchten Übungsgruppenleiter und die Anzahl der Räume geplant werden.

Räume und Mitarbeiter für Übungsveranstaltungen zuteilen
Weil die Verfügbarkeit der zwei Ressourcen Gruppenleiter und Räume noch unbekannt ist, ist die Aufgabe der nächsten Phase, diese zwei wichtigen Ressourcen für Übungen zuzuteilen. Die Aufgaben, Betreuer und Räume zu suchen, werden parallel vom Hauptverantwortlichen durchgeführt. Werden weitere Mitarbeiter benötigt, müssen sie von anderen Lehrstühlen des ganzen Fachbereichs rekrutiert werden. Weitere Räume können sogar an der ganzen Uni gesucht werden. Für viele Übungsveranstaltungen von Grundvorlesungen stehen nicht immer genug wissenschaftliche Mitarbeiter zur Verfügung. Deswegen werden sehr häufig studentische Übungsgruppenleiter gesucht. Dieser Vorgang wird nicht nur von dem Hauptverantwortlichen, sondern auch von der Verwaltungsabteilung für studentische Hilfskräfte an der Uni durchgeführt. Nach der Suchphase wird die Anfangsplanung (Gruppengröße, Gruppenanzahl) den verfügbaren Ressourcen angepasst.

Übungsplanung festlegen
Am Ende der Übungsplanung stehen der Termin, der Raum, der Übungsgruppenleiter, die Gruppengröße und die maximale Anzahl der Gruppen fest. Die Übungsplanungsphase ist damit abgeschlossen.

Verfeinerung „Teilnehmer einteilen“
Hier wird die Einteilung der Studenten auf die einzelnen Übungsgruppen genauer beschrieben.

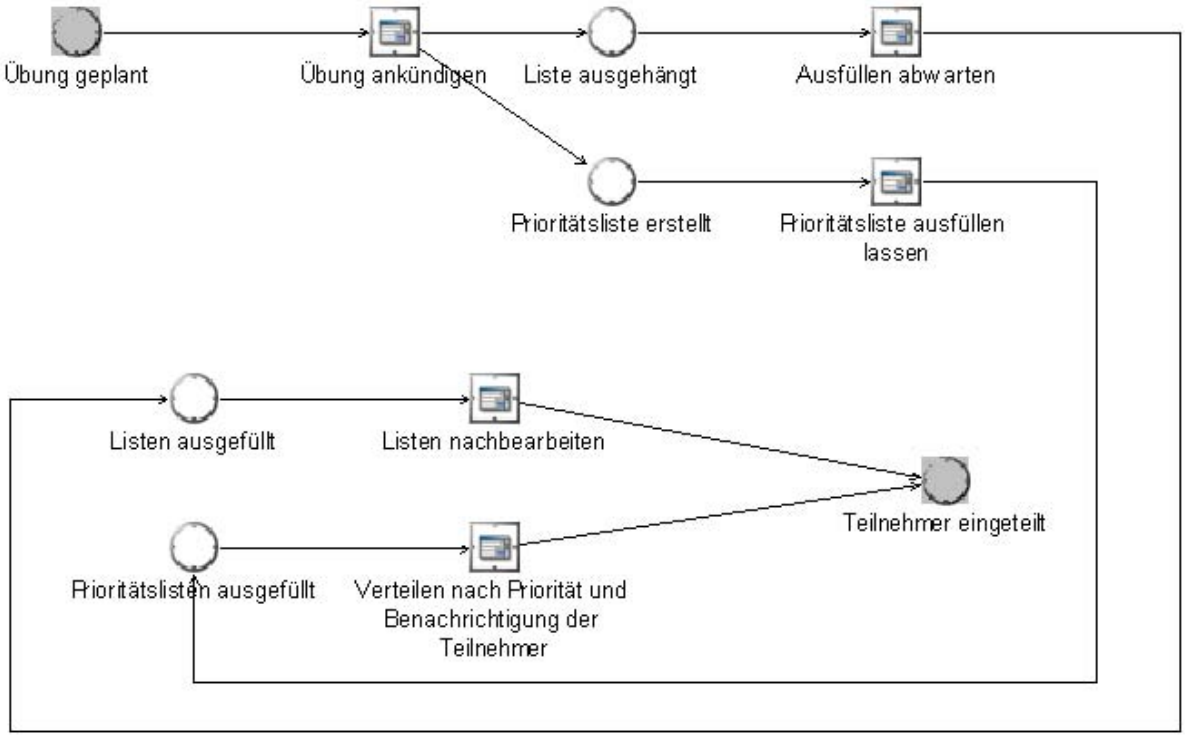


Abbildung 7-5: Ablauf der Einteilung von Teilnehmern

Übung ankündigen
Diese Phase beginnt mit der Ankündigung der Vorlesung. Wie dies geschieht obliegt einzig dem Veranstalter. Er kann Daten zur Übung auf seiner Homepage veröffentlichen oder die Übung einfach nur zu Beginn der Vorlesung mündlich ankündigen.

Ausfüllen abwarten / Liste nachbearbeiten
Für die Anmeldung der Studenten haben sich zwei Methoden durchgesetzt. Bei der ersten werden während der Vorlesung einige Listen herumgegeben, in denen sich jeder Student zu mehreren Übungsgruppen mit Angabe einer Priorität eintragen kann. Diese Listen werden dann wieder eingesammelt und später von Hand nachbearbeitet, da eine etwa gleiche Verteilung der Studenten auf die Übungsgruppen nicht zu erwarten ist. Den Teilnehmern wird danach ihre tatsächlich zugewiesene Übungsgruppe mitgeteilt. Dies geschieht durch einen Aushang am Schwarzen Brett.

Prioritätsliste ausfüllen / Benachrichtigung der Teilnehmer
Bei der zweiten Methode werden Listen an einem schwarzen Brett ausgehängt. In diese können sich die Studenten dann eintragen. Da hier die Anzahl der Plätze in den Übungsgruppen direkt ablesbar ist, ist eine manuelle Nachbearbeitung meist nicht erforderlich. Weiterhin wissen die Studenten direkt, welcher Übungsgruppe sie zugeordnet sind, so dass an dieser Stelle eine Benachrichtigung der Studenten entfallen kann.

Verfeinerung „Übung durchführen“
Dieses Modell beschreibt die Durchführung einer Übungsveranstaltung über einen Zeitraum von einem Semester.

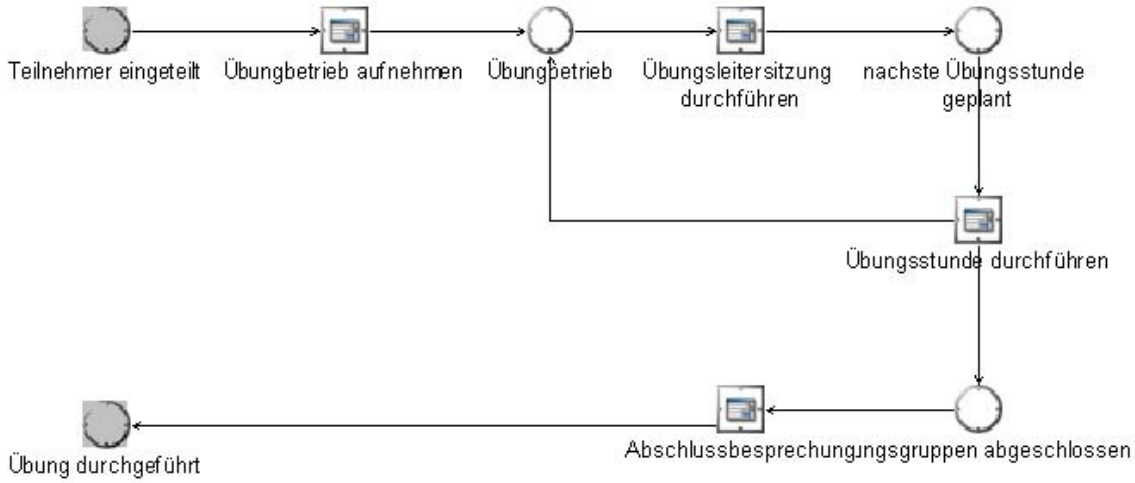


Abbildung 7-6: Ablauf der Durchführung einer Übungsveranstaltung

Übungsbetrieb aufnehmen
Nachdem die Teilnehmer eingeteilt wurden, kann der regelmäßige Übungsbetrieb aufgenommen werden. Die letzten Vorbereitungen für die regelmäßige Übung werden gemacht. An diesem Vorgang sind der Hauptverantwortliche, die jeweiligen Übungsleiter und eventuell auch die Veranstalter beteiligt. Damit sind die letzten Koordinationen für die nachfolgende Phase durchgeführt.

Der Übungsbetrieb und die Übungsleitersitzung
Danach fängt der Regelbetrieb an. Zunächst versammeln sich die Übungsleiter und besprechen während einer Sitzung die Aufgaben der nächsten Übungsstunde. Die Übungsaufgaben müssen eventuell noch eingetippt und ausgedruckt werden. Hiernach sind die Übungsleiter auf das Vortragen der Aufgaben und auf Rückfragen seitens der Übungsgruppenteilnehmer vorbereitet.

Die (wöchentliche) Übungsgruppe
Die Übungsgruppe wird üblicherweise ein Mal pro Woche durchgeführt. Jeweils ein Übungsleiter leitet eine Übungsgruppensitzung, welche üblicherweise etwa eineinhalb Stunden dauert. Der Übungsbetrieb erstreckt sich über ein Semester, danach gilt die Übungsgruppe allgemein als abgeschlossen. Das Ziel ist die Lösung der Aufgaben vorzutragen und vielleicht eine Musterlösung auszuteilen. In einigen Veranstaltungen wird auch die Anwesenheit kontrolliert, welche dann relevant für die spätere Scheinvergabe ist.

Die (optionale) Abschlussbesprechung
Am Ende findet gegebenenfalls eine Abschlussbesprechung statt. Teilweise findet sogar ein Treffen aller Übungsgruppenleiter und des Hauptverantwortlichen statt. Die Dauer dieser Besprechung ist nicht genauer zu spezifizieren. Das Ziel ist, Erfahrungen und Anmerkungen auszutauschen.

7.1.4 Analyse und Modellierung der Geschäftsprozesse für das Prüfungsmanagement

In diesem Kapitel wird der Ist-Zustand der Vorgänge beim Prüfungsmanagement beschrieben. Die Informationen stammen zum größten Teil aus persönlichen Gesprächen mit Herrn Hohmann vom Zentralen Prüfungsamt (ZPA), der für die Studiengänge Ingenieurinformatik und Kerninformatik zuständig ist. Aber auch Herr Hillemann vom ZPA war für unsere Fragen offen und für einen detaillierten Einblick in die Geschäftsprozesse des ZPA unverzichtbar.

Es existieren verschiedene Anmeldeformulare für Vordiploms- bzw. Hauptdiplomsprüfungen und zusätzlich die Unterscheidung zwischen Informatik und Angewandter Informatik. Darüber hinaus gibt es verschiedene Diplomprüfungsordnungen aus verschiedenen Jahren. So gibt es für den Bereich Informatik die DPO 1996 und die DPO 2001. Für den Bereich Angewandte Informatik gibt es Diplomprüfungsordnungen aus den Jahren 1993, 1997 und 2001.

Rollendefinition

Veranstalter	Derjenige, der die Prüfung veranstaltet.
ZPA	Ein Mitarbeiter des ZPA, der für die Administration der Studentendaten zuständig ist.
Student	Teilnehmer an einer Prüfung

Abbildung 7-7: Rollendefinition für das Prüfungsmanagement

7.1.4.1 Die mündliche benotete Prüfung

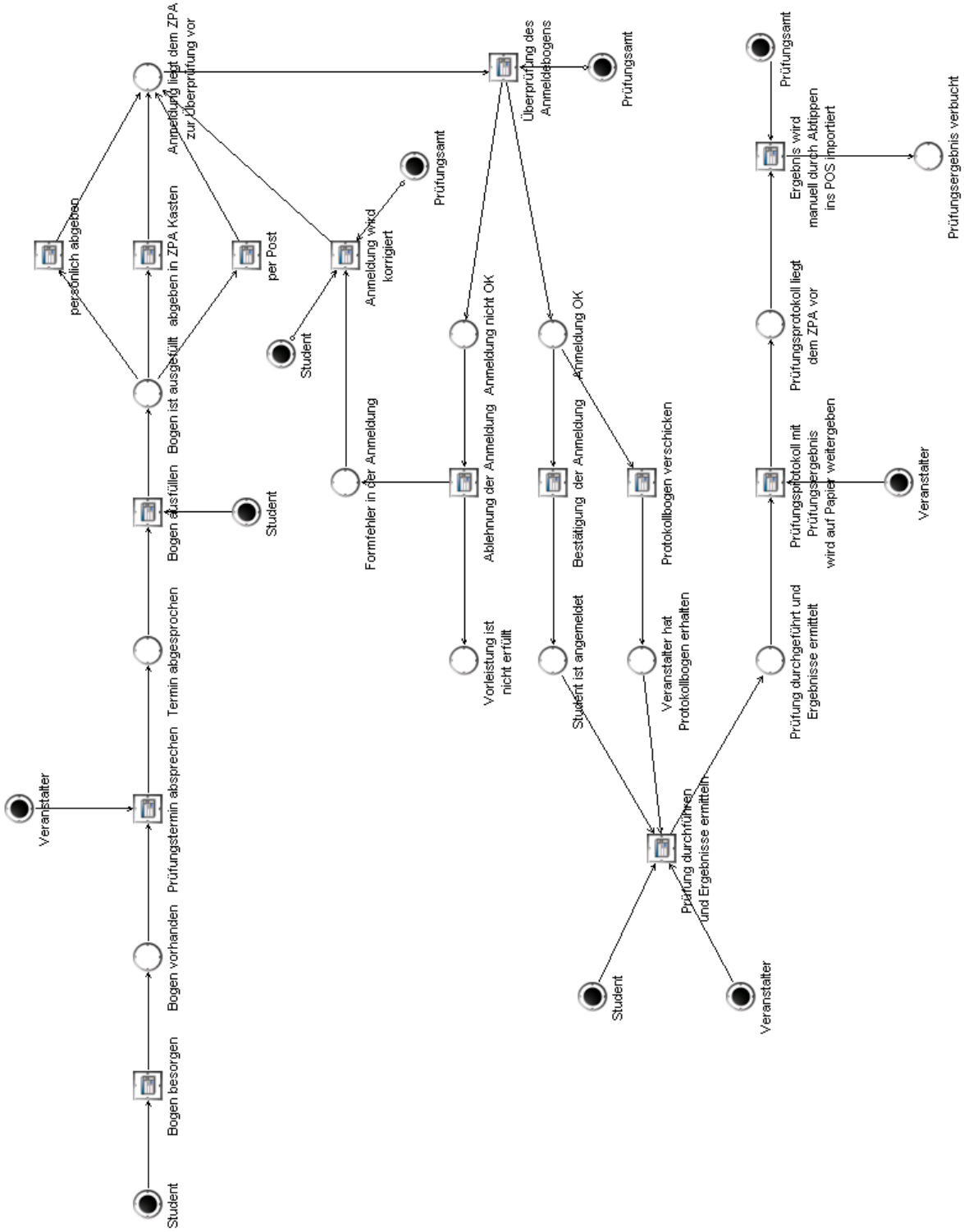


Abbildung 7-8: Ablauf einer mündlichen benoteten Prüfung

Terminvereinbarung und Prüfungsanmeldung
Möchte sich ein Student zu einer Prüfung anmelden, muss er sich zuerst den entsprechenden Anmeldebogen aus dem Zentralen Prüfungsamt (ZPA) besorgen. Der Anmeldebogen muss der DPO entsprechen, nach der der Prüfling studiert.

Student und Veranstalter sprechen gemeinsam den Prüfungstermin (z.B. persönlich, telefonisch oder auch per Email) ab, das Anmeldeformular muss nach erfolgter

Terminabsprache vom Veranstalter unterzeichnet werden. Zu beachten ist, dass das Anmeldeformular mindestens zwei Wochen vor der eigentlichen Prüfung am Prüfungsamt eingegangen sein muss.

Ist ein Prüfungstermin zustande gekommen, kann der Student den Anmeldebogen ausfüllen.

Zu den notwendigen Eintragungen gehören folgende Daten des Antragstellers:

- Matrikel-Nr.
- Name, Vorname
- Telefon / e-mail / Fax
- Daten zur Prüfung
- Zulassung von Zuhörern (ja/nein)
- Gruppenprüfung (ja/nein)
- ggf. Vorleistungen
- ggf. bisheriger Studienverlauf
- Prüfungsgebiete
- Prüfer
- Termin
- Unterschrift des Prüfers
- Datum und Unterschrift des Antragstellers

Nach dem Ausfüllen muss das Anmeldedokument an das ZPA weitergeleitet werden. Dazu hat der Student drei Möglichkeiten. Die Abgabe kann persönlich erfolgen, hierbei müssen die Öffnungszeiten des Prüfungsamtes beachtet werden. Im ZPA gibt es einen Briefkasten, in dem u.a. auch Prüfungsanmeldungsbögen eingeworfen werden können. Zuletzt hat der Student die Möglichkeit, seine Anmeldung auf dem Postweg zu verschicken.

Anmeldungsüberprüfung
Ein ZPA-Mitarbeiter prüft das eingegangene Anmeldeformular auf seine Korrektheit. Hier sind u.a. die verschiedenen Diplomprüfungsordnungen zu berücksichtigen. Es ist zu prüfen, ob etwaige Vorleistungen zu der Prüfung fehlen. Größtenteils geschieht dies manuell, da nur sehr wenige Vorleistungen im EDV-System eingetragen sind, so dass eine automatische Erkennung kaum möglich ist.

Falls das Anmeldeformular nicht korrekt ausgefüllt wurde oder eine Vorleistung fehlt, muss der Student benachrichtigt werden. Der Student wird in aller Regel schriftlich (d.h. per E-Mail) über eine Ablehnung seiner Prüfungsanmeldung informiert.
Bei Formfehlern muss der Student in der Regel dann noch einmal im ZPA erscheinen, um seinen Anmeldebogen zu korrigieren oder sogar einen neuen Anmeldebogen auszufüllen.

Ist der Anmeldebogen korrekt ausgefüllt und sind alle Vorleistungen erfüllt, trägt ein ZPA-Mitarbeiter die Daten in das EDV-System ein und der Student ist angemeldet.

Ein ZPA-Mitarbeiter verschickt den Prüfungsprotokollbogen an den Prüfer. Dieser Bogen bietet Platz für:

- das Prüfungsfach
- die Prüflingsinformationen
- das Protokoll
- das Prüfungsergebnis

Prüfungsergebnis, Prüfungsdurchführung und Ergebnisermittlung
Nachdem die Prüfung vollzogen ist, liegt das Ergebnis in Papierform auf dem Prüfungsprotokollbogen vor. Das Prüfungsprotokoll wird per Hauspost oder persönlich zum ZPA gebracht. Im ZPA wird das Prüfungsergebnis manuell ins System übernommen.

7.1.4.2 Die schriftliche benotete Prüfung

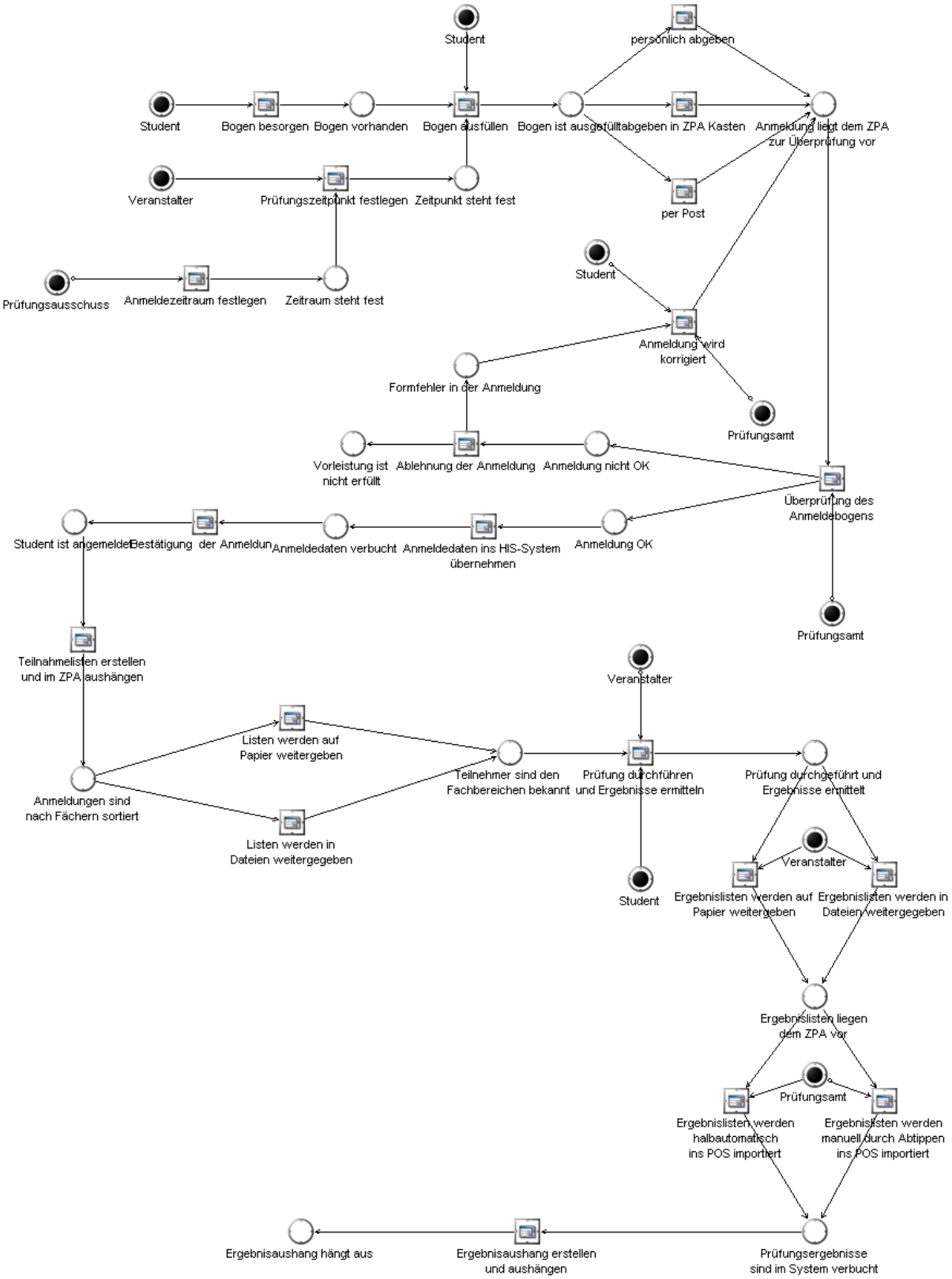


Abbildung 7-9: Ablauf einer schriftlichen benoteten Prüfung

Prüfungsanmeldung

Ähnlich der benoteten mündlichen Prüfung ist es für den Studenten notwendig, sich einen Anmeldebogen persönlich vom ZPA zu besorgen, auszufüllen und abzugeben. Anders als

bei der mündlichen Prüfung wird kein individueller Termin mit dem Veranstalter abgesprochen, vielmehr wird vom Veranstalter ein Prüfungstermin innerhalb des Prüfungszeitraums vorgegeben, an dem alle angemeldeten Studenten geprüft werden. Eine Unterschrift des Prüfers ist bei diesem Vorgang nicht notwendig. Die Abgabe des Anmeldebogens kann auf denselben Wegen erfolgen wie bei den mündlichen Prüfungen.

Anmeldungsüberprüfung
Sind die Anmeldebedingungen erfüllt (siehe mündliche Prüfung), werden die Daten von einem ZPA-Mitarbeiter ins EDV-System übernommen. Der Student erhält eine schriftliche Bestätigung über seine Anmeldung.

Ist der Anmeldezeitraum verstrichen, werden vom ZPA-Mitarbeiter Teilnehmerlisten erstellt und im ZPA ausgehangen. Diese Listen werden ebenfalls an den Prüfungsveranstalter weitergereicht, entweder in Listenform auf Papier oder aber als Excel-Datei. Die Daten werden entweder per Hauspost verschickt oder von einem autorisierten Mitarbeiter des veranstaltenden Lehrstuhls abgeholt.

Prüfungsdurchführung und Ergebnisermittlung
Sind dem Veranstalter alle Teilnehmer bekannt, kann er die Prüfung durchführen. Nach der Ergebnisermittlung füllt der Prüfer die Papierliste (bzw. die Excel-Datei) mit den Ergebnissen und lässt diese wiederum dem ZPA zukommen. Der ZPA-Mitarbeiter muss nun die Ergebnislisten in das EDV-System übertragen. Diese Arbeit ist größtenteils nur manuell zu lösen und extrem zeitaufwändig. Sind alle Ergebnisse verbucht, werden verbindliche Ergebnisaushänge vom ZPA-Mitarbeiter erstellt und ausgehängt.

7.1.4.3 Die schriftliche unbenotete Prüfung

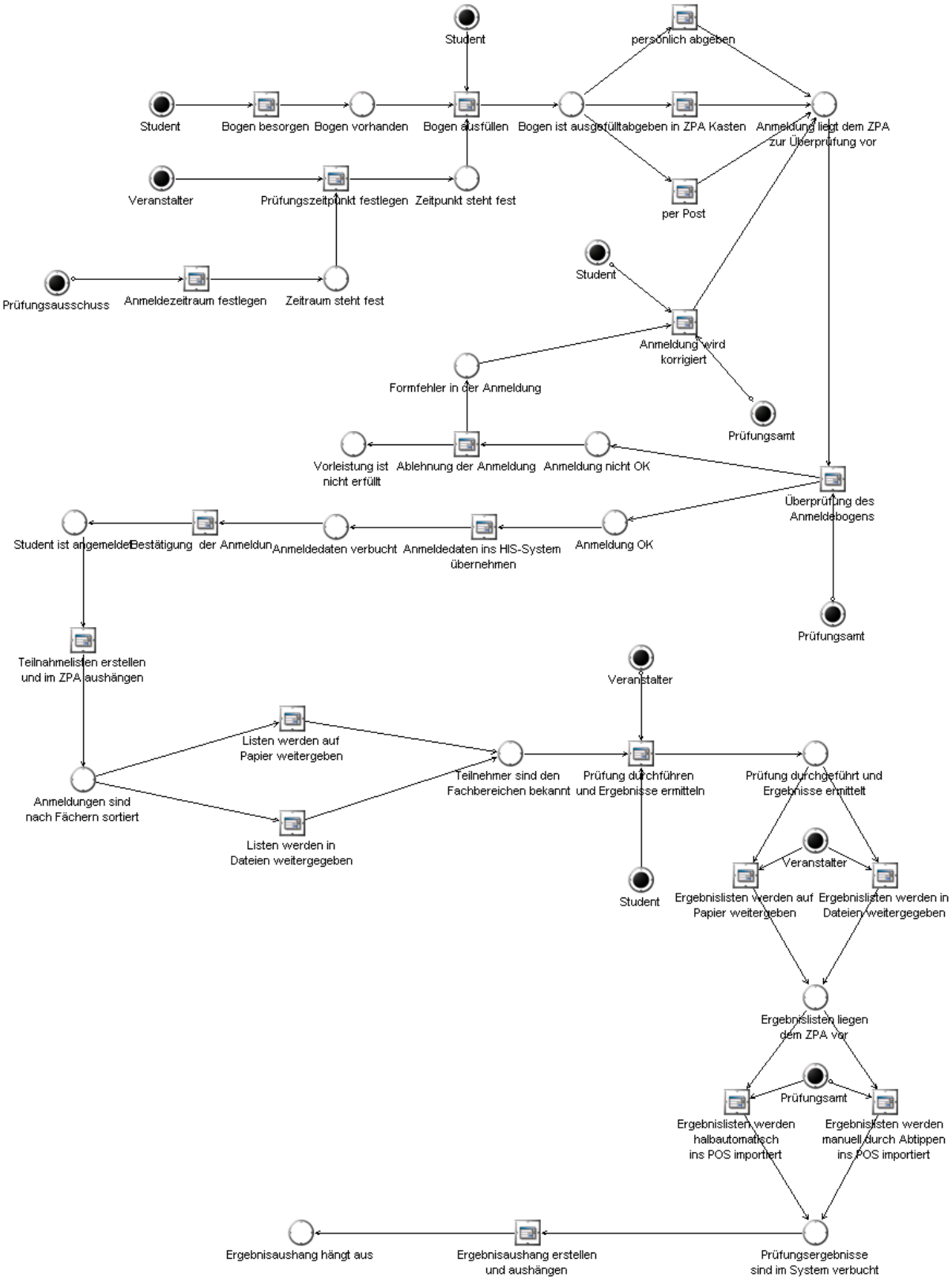


Abbildung 7-10: Ablauf einer schriftlichen unbenoteten Prüfung

Prüfungsanmeldung
Bei schriftlichen unbenoteten Prüfungen ist der Veranstalter auf keinen vom Prüfungsausschuss festgelegten Prüfungszeitraum angewiesen. Er ist selbst verantwortlich für Prüfungstermin und Anmeldezeitraum. Der Prüfer erstellt in der Regel einen Aushang mit den entsprechenden Daten und stellt den Studenten eine Liste zur Verfügung, in die sie sich eintragen können. Nach dem Eintrag in die Liste ist der Student angemeldet und die Prüfung kann durchgeführt werden.

Prüfungsdurchführung und Ergebnisermittlung
Nach der Ermittlung der Prüfungsergebnisse kann es unterschiedliche Wege geben, wie diese dem ZPA vermittelt werden:

- Der Prüfer druckt für alle Teilnehmer, die bestanden haben, den Schein aus und übergibt diese an die Studenten. Der Student ist daraufhin selbst dafür verantwortlich, dass der Schein zum ZPA gelangt.
- Der Fachbereich erstellt eine Liste aller erfolgreichen Teilnehmer und lässt diese dem ZPA zukommen. Der Student muss nicht persönlich beim ZPA erscheinen, um seine Leistung quittieren zu lassen.
- Die Prüfungsergebnisse werden wiederum größtenteils manuell in das EDV-System im ZPA durch einen ZPA-Mitarbeiter eingegeben.

7.1.4.4 Prüfungsabmeldung von mündlichen benoteten Prüfungen

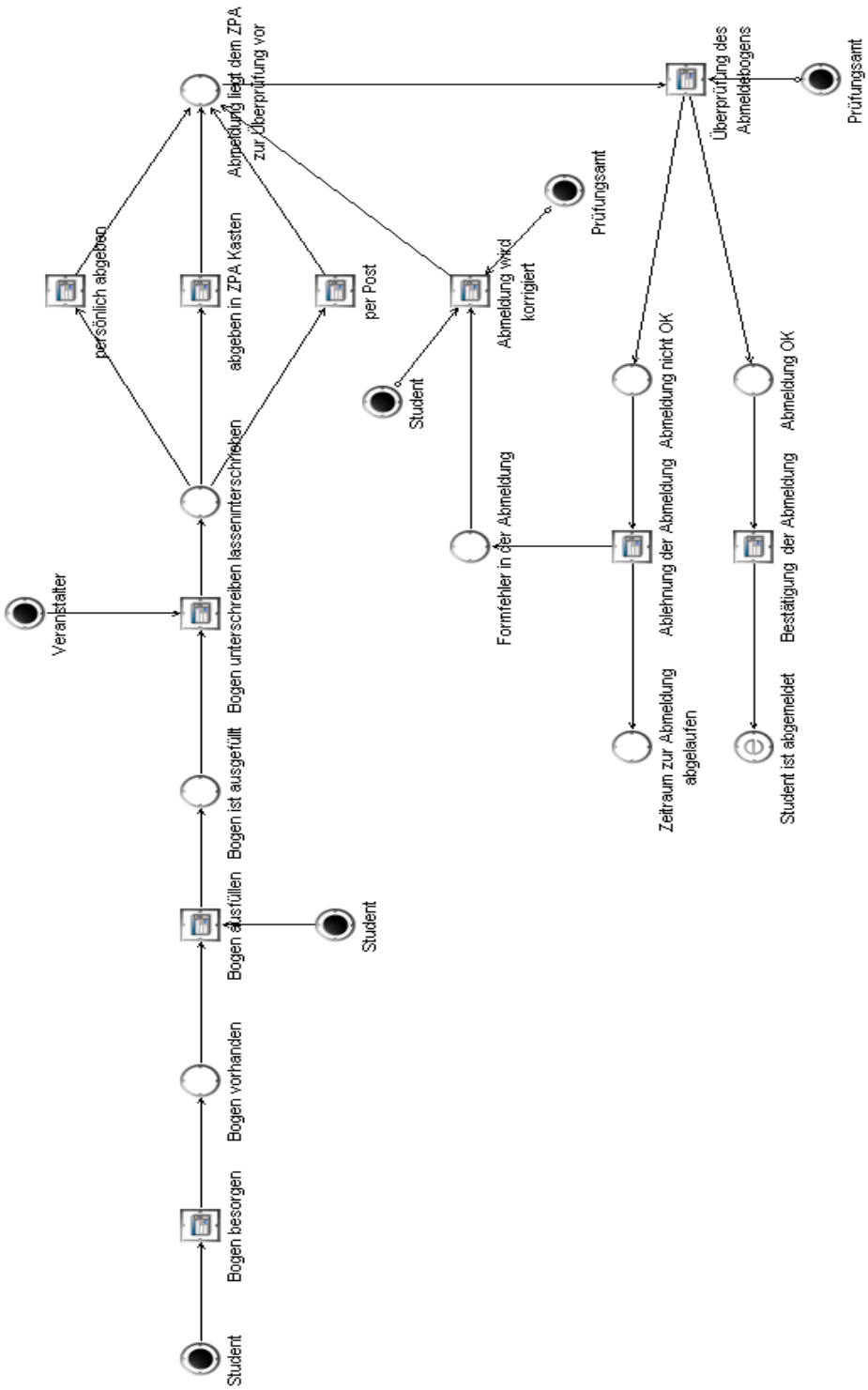


Abbildung 7-11: Prüfungsabmeldung von einer mündlichen benoteten Prüfung

7.1.4.5 Prüfungsabmeldung von schriftlichen benoteten Prüfungen

Die Prüfungsabmeldung für schriftliche, benotete Prüfungen entspricht größtenteils dem Vorgang der Abmeldung von mündlichen, benoteten Prüfungen. Es ist jedoch nicht notwendig, eine Unterschrift vom Prüfer einzuholen.

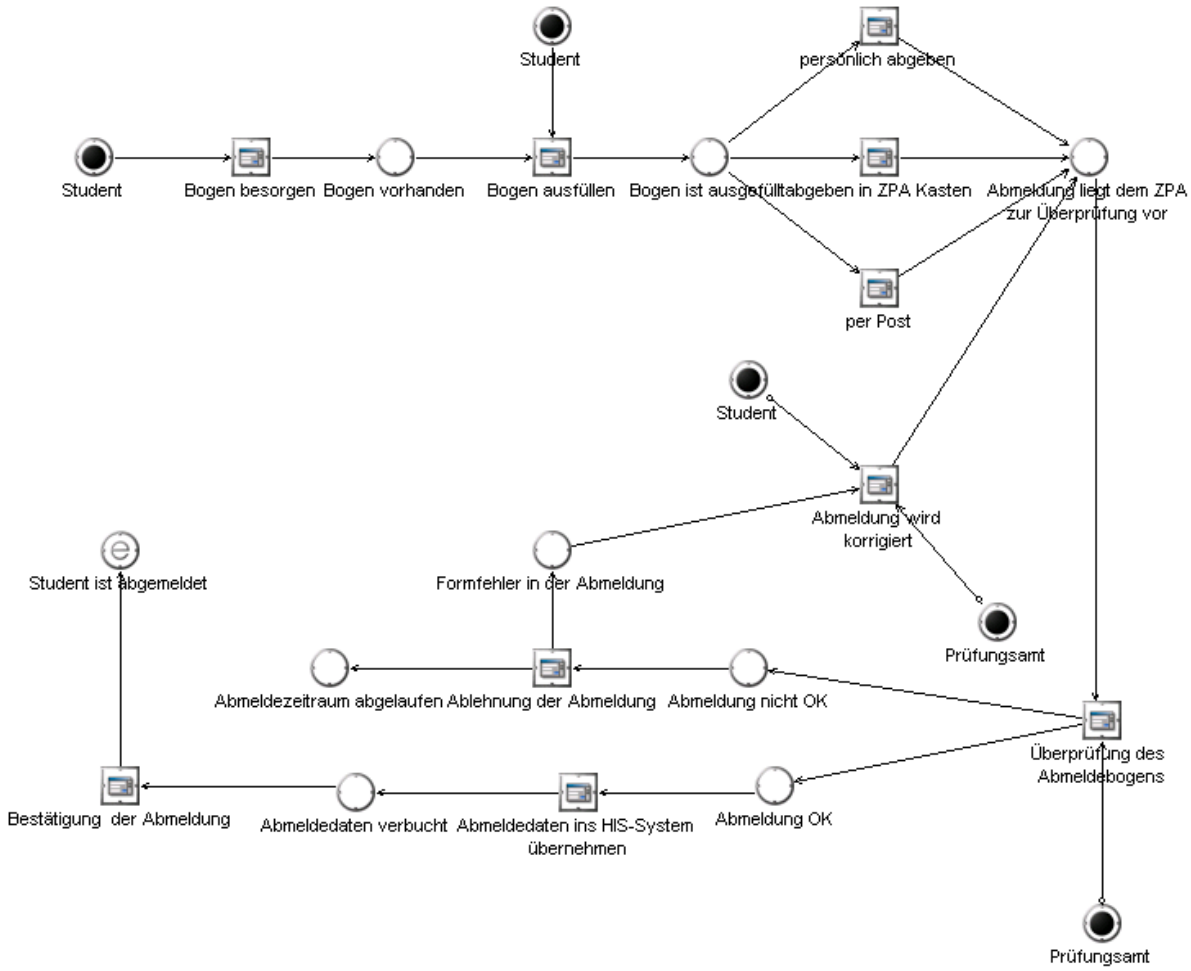


Abbildung 7-12: Abmeldung von einer schriftlichen benoteten Prüfung

Eine Abmeldung von einer Prüfung ist ein formloses Schreiben an das ZPA. Vom ZPA sind jedoch Vordrucke ausgelegt, die man dafür verwenden kann. Folgende Fälle sind zu unterscheiden:

- Bei einer unbenoteten Prüfung ist eine Abmeldung i.d.R. nicht erforderlich, da man diese so oft wie nötig wiederholen kann.
- Bei einer schriftlichen benoteten Prüfung geht die Abmeldung unterschrieben direkt beim ZPA-Mitarbeiter ein. Dies kann per Post, per ZPA-Briefkasten oder persönlich erfolgen. Die Abmeldung muss bis spätestens zwei Wochen vor der Prüfung erfolgt sein.
- Bei einer mündlichen Prüfung muss die Abmeldung zuvor vom Professor oder seiner Sekretärin unterschrieben und genehmigt werden. Dies muss bis spätestens eine Woche vor der Prüfung erfolgen.

Einzige Bedingung für die Abmeldung ist, dass die Abmeldefrist noch nicht abgelaufen ist. Sollte diese schon abgelaufen sein, kann ein Student nur noch durch ein ärztliches Attest von der Prüfung zurücktreten.

Hat ein Student sich nicht persönlich abgemeldet, geht ihm noch eine Abmeldebestätigung per Post zu, wenn die Abmeldung erfolgreich war.

7.2 Optimierung der Geschäftsprozesse

7.2.1 Soll-Beschreibung des Übungsgruppenmanagements

Dieser Abschnitt beschreibt den Ablauf einer Übungsgruppenveranstaltung in einem optimierten Modell. Zugrunde liegen das Übungsgruppenmanagementmodell von Herrn Dißman, sowie eigene Erfahrungen der Studenten aus dem Übungsbetrieb. Aufgrund von Ist-Zustand und Gesprächen mit Herrn Dißman wurde die Optimierung durchgeführt und dabei sowohl auf die Sicht der Veranstalter als auch der Studenten Rücksicht genommen.

7.2.1.1 Übersicht Übungsgruppenmanagement

Dieses Modell zeigt den Ablauf einer Übungsgruppenveranstaltung in einer Übersicht. Wichtige Abläufe werden in separaten Modellen näher beschrieben. Einige Teile ändern sich nur wenig im Vergleich zum Ist-Zustand, da bestimmte Prozessabläufe schon durch die Hierarchie der Verantwortlichkeit vorgegeben sind.

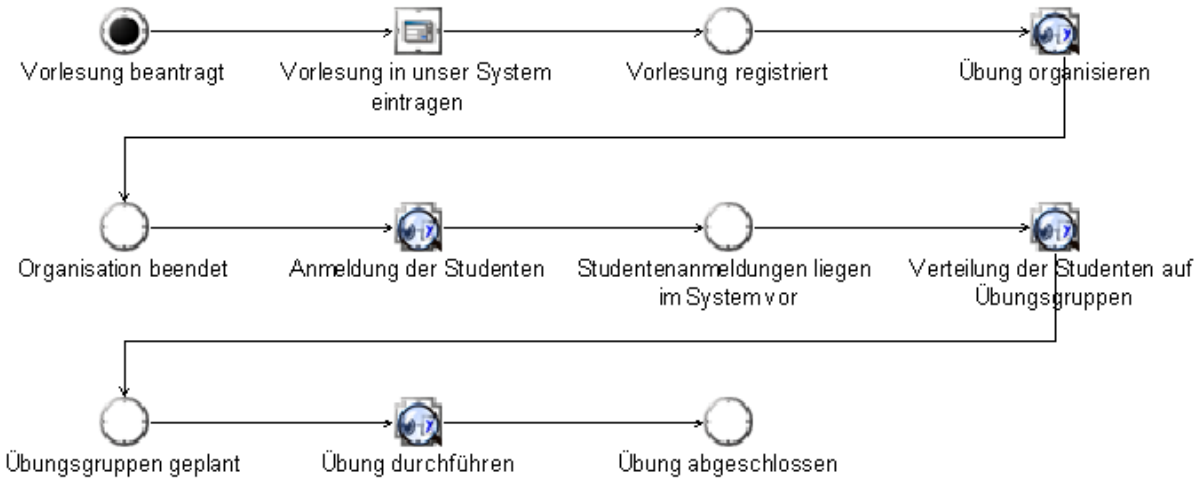


Abbildung 7-13: Übersicht über das Übungsmanagement

7.2.1.2 Beschreibung der einzelnen Phasen

Vorlesung beantragen

Zunächst wird eine Vorlesung, für die eine Übungsgruppenveranstaltung stattfinden soll, von dem Veranstalter beantragt. Nachdem die Vorlesung genehmigt wurde, kann sie ins Vorlesungsverzeichnis aufgenommen und in das Übungsgruppenmanagement-System eingetragen werden. Zu diesem Zeitpunkt sind noch keine Termine der einzelnen Übungsgruppen bekannt. Allerdings erhält man eine Übersicht über alle Vorlesungen, für die eine Übung geplant ist.

Vorlesung in das System eintragen

Die genehmigte Vorlesung wird im System registriert. Dies erledigt der Veranstalter. Angedacht ist hierbei die Einstellung von Informationen zur Vorlesung und der Übungsveranstaltung. Der Benutzer erhält dadurch eine Auflistung aller geplanten Veranstaltungen.

Übung organisieren

An dieser Stelle wird die Übung organisiert. Für jeden Fachbereich existiert ein Hauptverantwortlicher, der für die Durchführung und Planung aller Übungen verantwortlich ist. Der Verantwortliche schätzt in dieser Phase die Teilnehmerzahl und bestimmt damit die Anzahl der Räume und Mitarbeiter, die für diese Übungsveranstaltung notwendig sind. Dies

wird für jede Übungsveranstaltung durchgeführt. Der genaue Ablauf wird in dem Modell „Verfeinerung Übung organisieren“ erläutert.

Anmeldung der Studenten
Nachdem die Organisation abgeschlossen ist und die Räume und Termine bekannt sind, werden sie durch das System bekannt gegeben. Daraufhin erfolgt eine Anmeldephase, in der sich die Studenten über das System für eine Übungsveranstaltungen anmelden können. Dabei werden Terminprioritäten vom Studenten vergeben. Nach Anmeldeschluss liegen die Anmeldedaten dem System vor, sind aber noch nicht auf die Termine verteilt. Der genaue Ablauf wird im Modell „Verfeinerung Anmeldung der Studenten“ beschrieben.

Verteilung der Studenten auf Übungsgruppen
Ist die Anmeldephase abgeschlossen, können sich keine Studenten mehr anmelden. Die gewonnenen Anmeldedaten werden nun dazu genutzt um die Studenten gleichmäßig auf die einzelnen Termine zu verteilen. Dabei werden die vergebenen Prioritäten berücksichtigt. Dies erledigt ein Algorithmus, der von dem Verantwortlichen angestoßen wird. Danach besteht immer noch die Möglichkeit der manuellen Nachbearbeitung. Der genaue Ablauf ist im Modell „Verfeinerung Verteilung der Studenten“ zu finden.

Übung durchführen
Nun können die Übungsgruppen durchgeführt werden. In jeder Woche des Semesters findet eine Übungsgruppenleiterbesprechung statt, in der die nächste Übungsstunde geplant wird. Dazu werden Übungszettel im System verfügbar gemacht. Diese werden dann von den Studenten bearbeitet und die Ergebnisse (wenn möglich) ins System zurückgegeben. Die Abgaben werden korrigiert und die erreichten Punkte ins System übertragen. Diese Arbeiten erledigen die Übungsgruppenleiter. Zudem werden Musterlösungen im System freigegeben. Am Ende des Semesters ist die Gesamtpunktzahl jedes Studenten im System gespeichert. Nun werden vom System Übungsscheine erstellt. Die Verarbeitung wird vom Veranstalter angestoßen und überprüft. Damit ist die Übungsveranstaltung abgeschlossen. Eine genauere Beschreibung des Ablaufs sind in den Modellen „Verfeinerung Übung durchführen“ und „Verfeinerung Übung findet statt“ zu finden.

Verfeinerung „Übung organisieren“
Dieses Modell beschreibt den genauen Ablauf der Organisation einer Übungsveranstaltung.

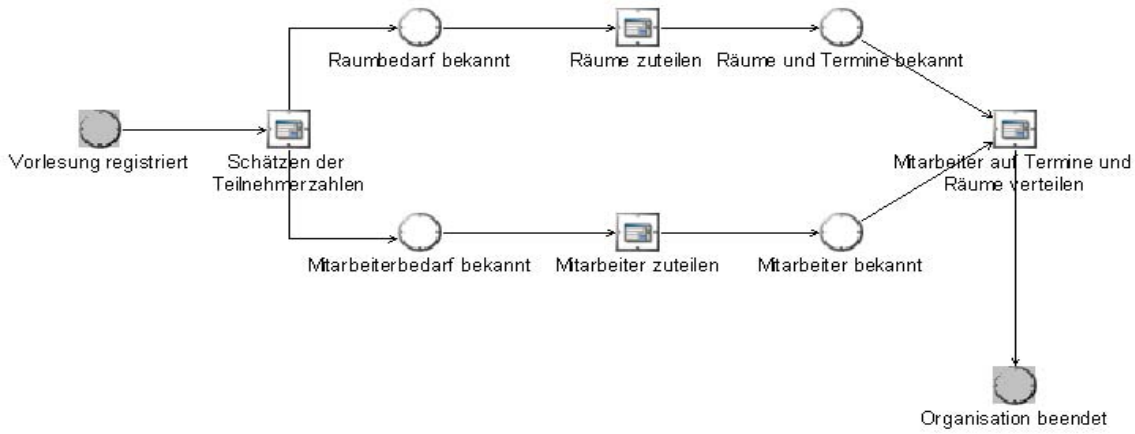


Abbildung 7-14: Ablauf der Organisation einer Übungsveranstaltung

Schätzen der Teilnehmerzahlen
Nachdem die Vorlesung im System registriert wurde, wird die Übung organisiert. Hierzu wird zunächst auf Basis der Teilnehmerzahlen der Übungsveranstaltung in vorherigen Semestern zu der entsprechenden Vorlesung die voraussichtliche Teilnehmerzahl vom Hauptverantwortlichen geschätzt. Dabei werden auch die aktuellen Anfänger- und Studierendenzahlen berücksichtigt. Aufgrund dieser geschätzten Zahl legt der Hauptverantwortliche den Raum- und Mitarbeiterbedarf für die Übungsveranstaltung fest und pflegt sie in das System ein.

Räume zuteilen
In dem System können dann aus einem Pool von Räumen, die dem Fachbereich zur Verfügung stehen und die vorher einmal inklusive ihrer Kapazität in das System eingegeben werden müssen, bestimmte Räume zu bestimmten Terminen der Übungsveranstaltung zugeteilt werden. Dabei muss darauf geachtet werden, dass die Termine sich nicht mit der Vorlesung, zu der die Übungsveranstaltung gehört, überschneiden. Des Weiteren sollte vermieden werden, dass die Termine mit anderen Vorlesungen kollidieren, die typischerweise von den Teilnehmern gehört werden. Die Zuteilung der Räume kann automatisch durch das System durchgeführt und hinterher gegebenenfalls manuell nachbearbeitet werden.

Mitarbeiter zuteilen
Parallel zur Raumzuweisung werden studentische und wissenschaftliche Mitarbeiter der Übungsveranstaltung, nach einer Absprache mit den betroffenen Personen, zugeteilt.

Mitarbeiter auf Termine und Räume verteilen
Nachdem die erforderlichen Räume und Mitarbeiter feststehen, werden die Mitarbeiter auf die einzelnen Termine verteilt, auch hierfür ist eine Absprache mit den jeweiligen Mitarbeitern erforderlich. Die Mitarbeiterzuteilung kann sich über mehrere Wochen erstrecken, da ihre Zuordnung sehr aufwändig ist. Danach ist die Organisation beendet.

Verfeinerung „Anmeldung der Studenten“
Dieses Modell beschreibt den Ablauf der Anmeldung von Studenten an das System.

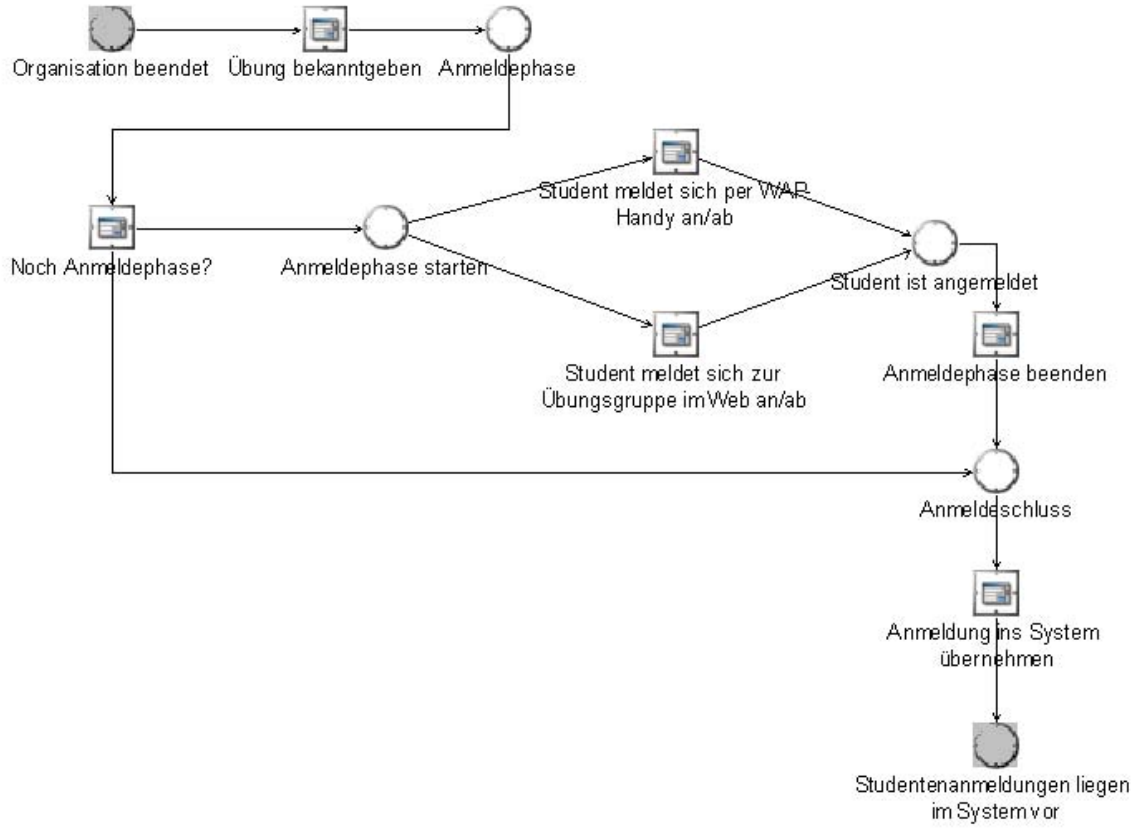


Abbildung 7-15: Ablauf der Anmeldung von Studenten

Übung bekannt geben
Nachdem die Organisationsarbeit beendet ist, wird die Übung angekündigt, d.h. Termin, Raum, Betreuer und Gruppengröße jeder Gruppe werden z.B. auf einer Webseite bekannt gegeben. Allerdings können sich diese Daten bei Bedarf noch ändern, wenn z.B. ein Tag in der Woche besonders gefragt ist.

Danach haben die Studenten, die an der Übungsveranstaltung teilnehmen möchten, zwei Möglichkeiten, sich für die Übung anzumelden. Die erste Möglichkeit ist, sich per WAP-Mobiltelefon anzumelden. Die zweite, sich im Web für eine Übungsgruppe anzumelden. Das heißt, die Studenten müssen ein Onlineformular ausfüllen. Dieses Formular enthält den Namen, Matrikelnummer, Priorität usw. Die Studenten müssen jedes Eingabefeld vollständig ausfüllen, sonst wird das Formular nicht angenommen. Der Student kann natürlich vor dem Senden das Formular zurücksetzen. Nach dem Absenden eines vollständig ausgefüllten Formulars werden die Daten im System aufgenommen und gespeichert. Diese Daten stehen für die weitere Verarbeitung zur Verfügung. Damit ist der Anmeldevorgang abgeschlossen.

Verfeinerung „Verteilung der Studenten auf Übungsgruppen“
Dieses Modell beschreibt den genauen Ablauf der Verteilung von Studenten auf die einzelnen Übungsgruppen.

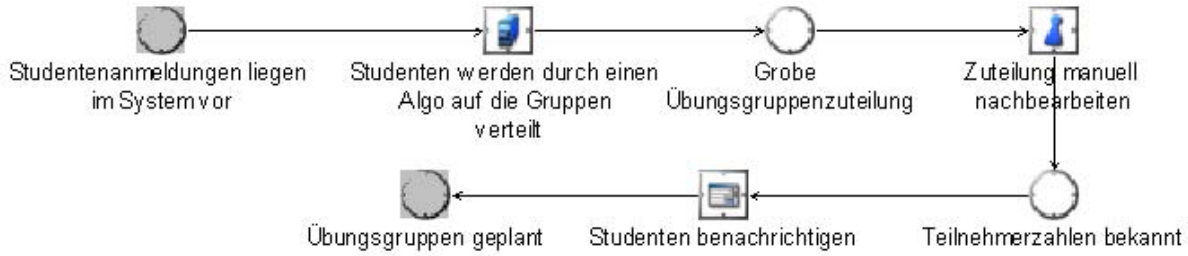


Abbildung 7-16: Ablauf der Verteilung von Studenten auf die einzelnen Übungsgruppen

Studenten werden auf die Gruppen verteilt
Nachdem die Anmeldungsphase abgeschlossen ist, liegen dem System alle benötigten Daten der Studenten vor, also Namen, Matrikelnummern und die angegebenen Terminprioritäten.

Jetzt werden alle Studenten, die sich für eine bestimmte Übungsveranstaltung angemeldet haben, gleichmäßig auf die vorhandenen Termine verteilt. Dieser Vorgang wird vom System ausgeführt und sollte von dem Verantwortlichen angestoßen werden. Der Algorithmus berücksichtigt dabei die vom Studenten angegebenen Prioritäten soweit wie möglich. Damit hat jeder Student die gleiche Chance der von ihm favorisierten Übungsgruppe zugeteilt zu werden, unabhängig vom Zeitpunkt der Anmeldung. Die Verteilung kann für jede Veranstaltung getrennt angestoßen werden.

Zuteilung manuell nacharbeiten
Nach der automatischen Zuteilung der Studenten auf die vorhandenen Termine, kann die Verteilung noch manuell nachbearbeitet werden. Der Hauptverantwortliche kann somit nachträglich Studenten in andere Übungsgruppen verschieben. Diese Nachbearbeitung ist jederzeit möglich. Damit sind die Übungsgruppen geplant.

Verfeinerung „Übung durchführen“
In diesem Modell wird beschrieben, wie die Übungsgruppen durchgeführt werden. Es wird der Zeitraum eines Semesters betrachtet. Eine Verfeinerung ist im Modell „Verfeinerung Übung findet statt“ zu finden.

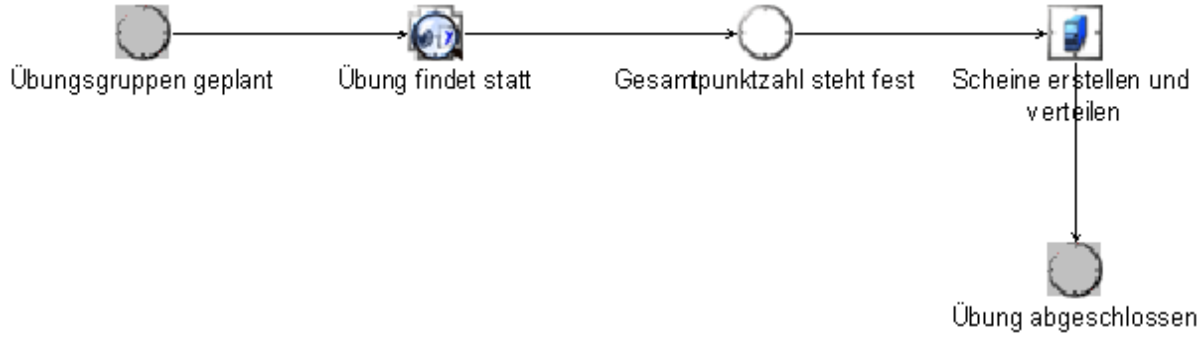


Abbildung 7-17: Verfeinerung „Übung durchführen“

Übung findet statt
Nachdem die Übungsgruppe geplant ist, finden im Allgemeinen ab der zweiten Vorlesungswoche wöchentliche Übungssitzungen statt. Weitere Details werden unter „Verfeinerung Übung findet statt“ erläutert.

Scheine erstellen und verteilen

Am Ende des Semesters, nach Durchführung der letzten Übungssitzung, steht die Gesamtpunktzahl der Teilnehmer, die von den Übungsgruppenleitern während der Durchführung der Übung wöchentlich aktualisiert wurde, fest und ist im System erfasst. Dieses kann dann automatisch entscheiden, welcher Teilnehmer die Scheinkriterien erfüllt hat, und ermöglicht es dem jeweiligen Übungsgruppenleiter die Scheine auszudrucken. Hierzu ist es nicht notwendig die Daten des Teilnehmers erneut einzugeben, da Name und Matrikelnummer dem System bereits vorliegen. Allerdings können die Übungsgruppenleiter manuell in die Scheinvergabe eingreifen, falls dies nötig sein sollte. Nachdem die Scheine erstellt wurden können sie den Teilnehmern ausgehändigt werden. Die Liste der Teilnehmer die einen Schein erworben haben kann automatisch erstellt und dem Prüfungsamt übermittelt werden. Nach der Scheinvergabe gilt die Übung als beendet.

Verfeinerung „Übung findet statt“
Hier ist der genaue Ablauf einer Übungsveranstaltung dargestellt. Der betrachtete Zeitraum ist ein Semester.

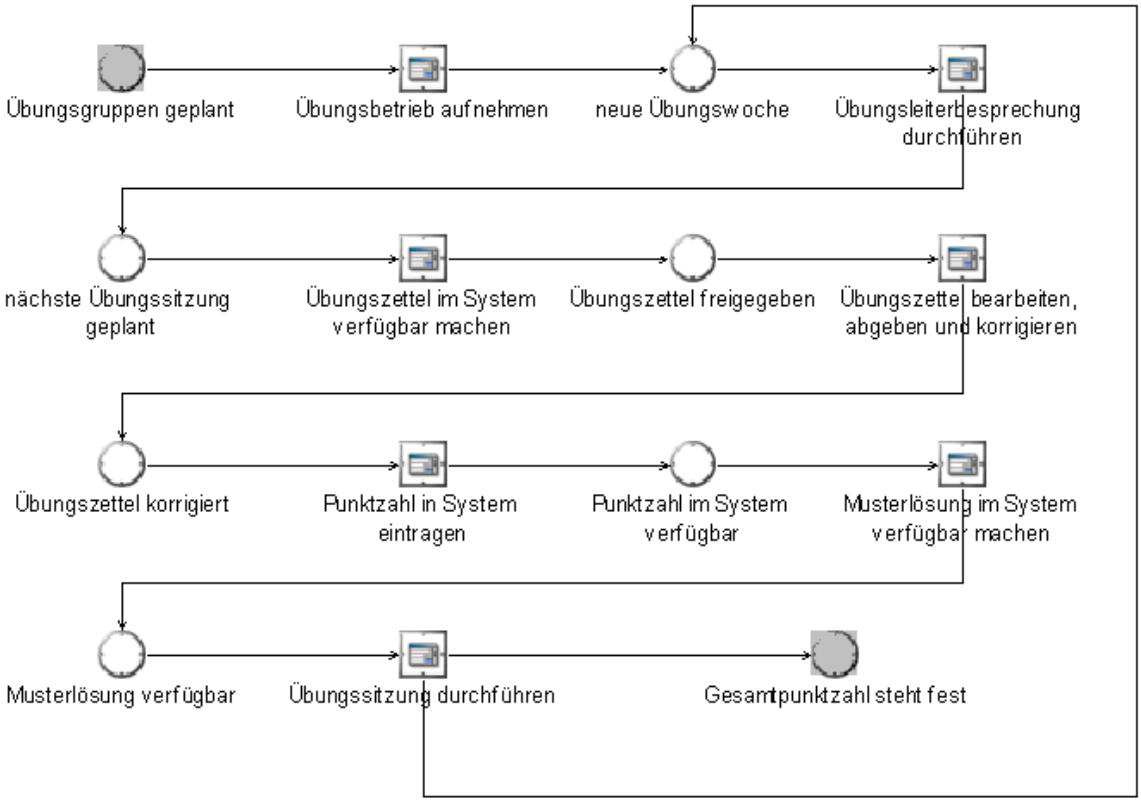


Abbildung 7-18: Ablauf einer Übungsveranstaltung

Übungsbetrieb aufnehmen (die letzten Vorbereitungen)
Nachdem die Übungsgruppenplanung gemacht wurde, kann der regelmäßige Übungsbetrieb aufgenommen werden. Hier werden die letzten Vorbereitungen für die regelmäßigen Übungen getroffen. An diesem Vorgang sind der Hauptverantwortliche, die jeweiligen Übungsgruppenleiter und eventuell auch die Veranstalter beteiligt. Der Vorgang kann unter Umständen mehrere Wochen dauern.

Die Übungsbesprechung
Am Anfang jeder Übungswoche findet eine Übungsbesprechung statt, bei der die Aufgaben für die nächste Sitzung diskutiert werden. Damit ist die nächste Übungsgruppensitzung geplant und die Übungsleiter sind auf das Vortragen der Aufgaben und auf Rückfragen seitens der Übungsgruppenteilnehmer vorbereitet.

Aufgaben ins System eingeben
Nach der Besprechung werden die Übungsaufgaben über unser System verfügbar gemacht und der Zugriff für die Übungsteilnehmer freigegeben. Die Aufgaben werden von mindestens einem Übungsgruppenleiter am Computer erstellt. Je nach Art der Aufgabe ist die Erstellung unterschiedlich aufwändig. Während z.B. Multiple-Choice Aufgaben recht schnell erstellt sind, kann eine komplexe Aufgabenstellung schon mal sehr aufwändig sein. In jedem Fall wird der Druckvorgang der Übungszettel eingespart. Damit unterstützt das System die Forderung nach „möglichst papierlosen Prozessen“. Die einzelnen Übungsgruppenteilnehmer haben nun Zeit, diese Übungsaufgabe zu bearbeiten und nach der Fertigstellung abzugeben. Die Bearbeitung wird in den meisten Fällen am Computer erledigt. Besondere Abgaben erfordern unter Umständen das Scannen einer Unterlage. Möglich sind hier private Scanner aber auch Scanner, die für diesen Zweck beim jeweiligen Lehrstuhl verfügbar sind. Der Prozess wird hier deutlich vereinfacht und für alle Beteiligten bequemer. Zusätzlich wird während dieses Vorgangs im Vergleich zum Ist-Zustand viel Zeit eingespart.

Die Abgabe der Übungsaufgaben
Die Aufgaben werden je nach Aufgabenart entweder elektronisch oder vom Übungsgruppenleiter geprüft und gegebenenfalls korrigiert. Hier sollten Übungsgruppenleiter jeweils eine Aufgabe korrigieren (übungsgruppenübergreifend) und so den Prozess beschleunigen. Manche Arten von Aufgaben können sogar automatisch korrigiert werden. Mit Rechnerunterstützung kann dabei zusätzlich Zeit eingespart werden.

Punkte ins System eingeben
Die Zahl der Punkte, die die Übungsgruppenteilnehmer erreicht haben wird dann auf den neuesten Stand gebracht. Dies kann in Verbindung mit der automatischen Korrektur vollständig automatisiert werden. Eingaben per Hand werden damit auf ein Minimum reduziert. In jedem Fall wird der Vorgang vereinfacht und beschleunigt.

Musterlösung verfügbar machen
Nachdem die Aufgaben korrigiert wurden können optional die Musterlösungen ins System gestellt werden. Damit spart man Zeit innerhalb der eigentlichen Übung, in der sie üblicherweise ausgeteilt werden. Zusätzlich ist der komplette Vorgang papierlos.

Die (wöchentliche) Übung
Danach findet die eigentliche Übung statt, wobei die Übungsgruppen üblicherweise wöchentlich abgehalten werden. Diese werden von genau einem Übungsgruppenleiter betreut und dauern für gewöhnlich 1½ Stunden. Der gesamte Vorgang wiederholt sich regelmäßig und fängt wieder mit der Übungsbesprechung an.

Scheinvergabe
Das Ergebnis der Übung ist natürlich die Punktezahl jedes Teilnehmers, die automatisch aus den Vorgängen entsteht. Die Liste der Teilnehmer, die einen Schein erworben haben, kann ebenfalls sehr einfach erstellt und sogar automatisch zum Prüfungsamt geschickt werden.

7.2.2 Soll-Beschreibung für das Prüfungsmanagement

Das Prüfungsmanagement stellt den zweiten zentralen Teil dieser PG dar, in welchem das Augenmerk auf der Automatisierung bisher schwierig zu handhabender Geschäftsprozesse im Fachbereich Informatik liegt.

7.2.2.1 Die mündliche benotete Prüfung

Dieses Modell beschreibt den Prozess bei der Anmeldung zu mündlichen benoteten Prüfungen.

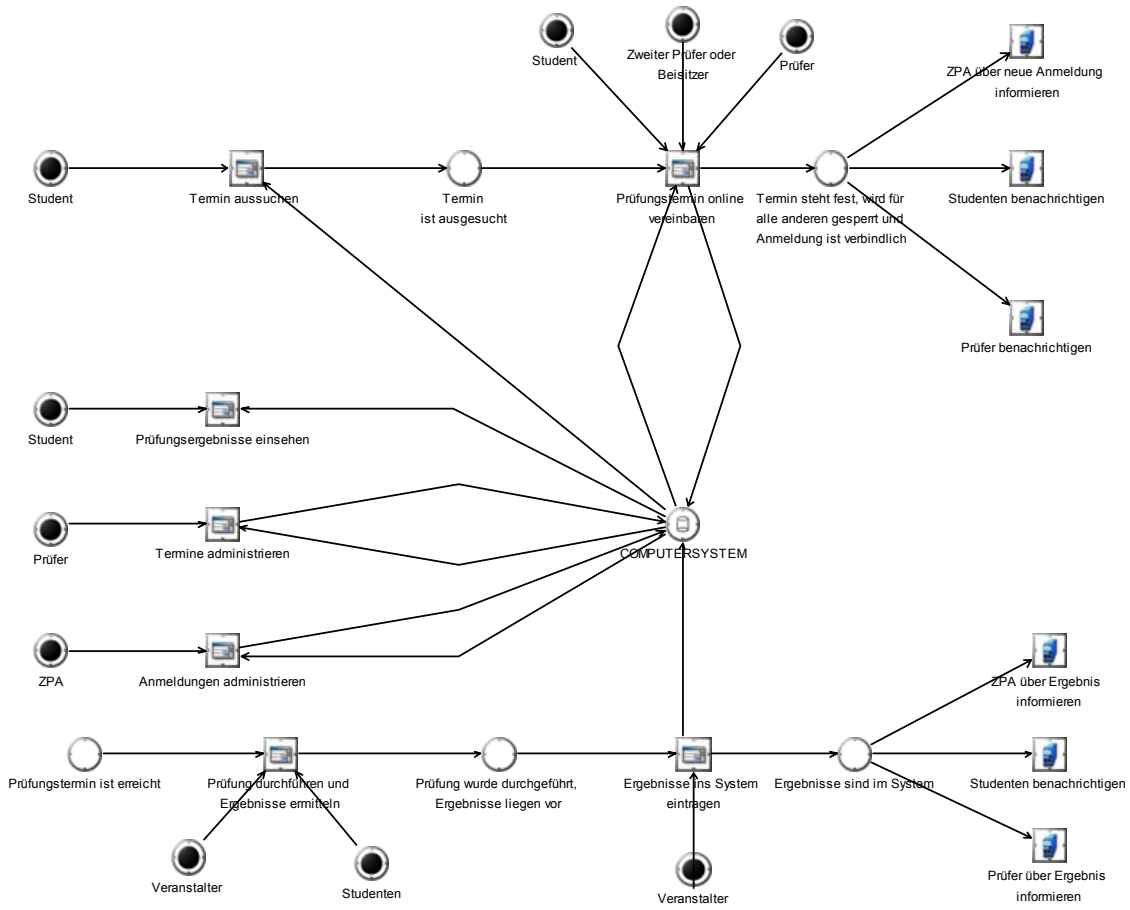


Abbildung 7-19: Anmeldung zu einer mündlichen benoteten Prüfung

Terminvereinbarung und Prüfungsanmeldung

Der Student erhält nach seinem Login eine Übersicht aller mündlichen Prüfungstermine der Prüfer, die das von ihm gewünschte Prüfungsgebiet prüfen. Er reserviert sich seinen persönlichen Wunschtermin, der dann auch aus der Prüfersicht zugänglich ist. Prüfer und Student können online Rücksprache halten und schließlich beidseitig dem Prüfungstermin zusagen. Somit ist der Prüfungstermin verbindlich im Computersystem verbucht. Der Student, der Prüfer und das ZPA erhalten eine Bestätigung über die Anmeldung und der vergebene Prüfungstermin wird für alle anderen Studenten gesperrt. Sind benötigte Vorleistungen nicht erfüllt, so wird dem Studenten die Anmeldung vom System verwehrt.

Prüfungsdurchführung und Ergebnisermittlung

Nach durchgeführter Prüfung und Notenermittlung trägt der Prüfer das Prüfungsergebnis in das System ein. Unmittelbar danach steht das Prüfungsergebnis im System für den Studenten, den Prüfer sowie das ZPA zum Abruf bereit. Der Student wird daraufhin automatisch benachrichtigt. Die eigentliche Note kann nur nach einem Login ins Computersystem über eine sichere Verbindung erfragt werden. Weiterhin geht eine Benachrichtigung sowohl an das ZPA als auch den Prüfer.

Systemabfragen

Der Student kann jederzeit seine Prüfungsergebnisse einsehen.

Prüfer haben die Möglichkeit, über eine Terminverwaltung des Systems ihre Prüfungstermine für Studenten freizugeben. Sie tragen dazu freie Prüfungstermine sowie die Fächer, in denen sie prüfen ein. Falls Prüfungsbedingungen erfüllt sein müssen, kann der Prüfer diese ebenfalls über das System eintragen.

Das ZPA kann zu jeder Zeit Anmeldungen administrieren. D.h. Anmeldungen können im Nachhinein noch abgelehnt werden, Benachrichtigungen an den Studenten erfolgen dann wiederum über das System.

7.2.2.2 Die schriftliche benotete Prüfung

Dieses Modell zeigt den Anmeldevorgang für schriftliche benotete Prüfungen in einer Übersicht.

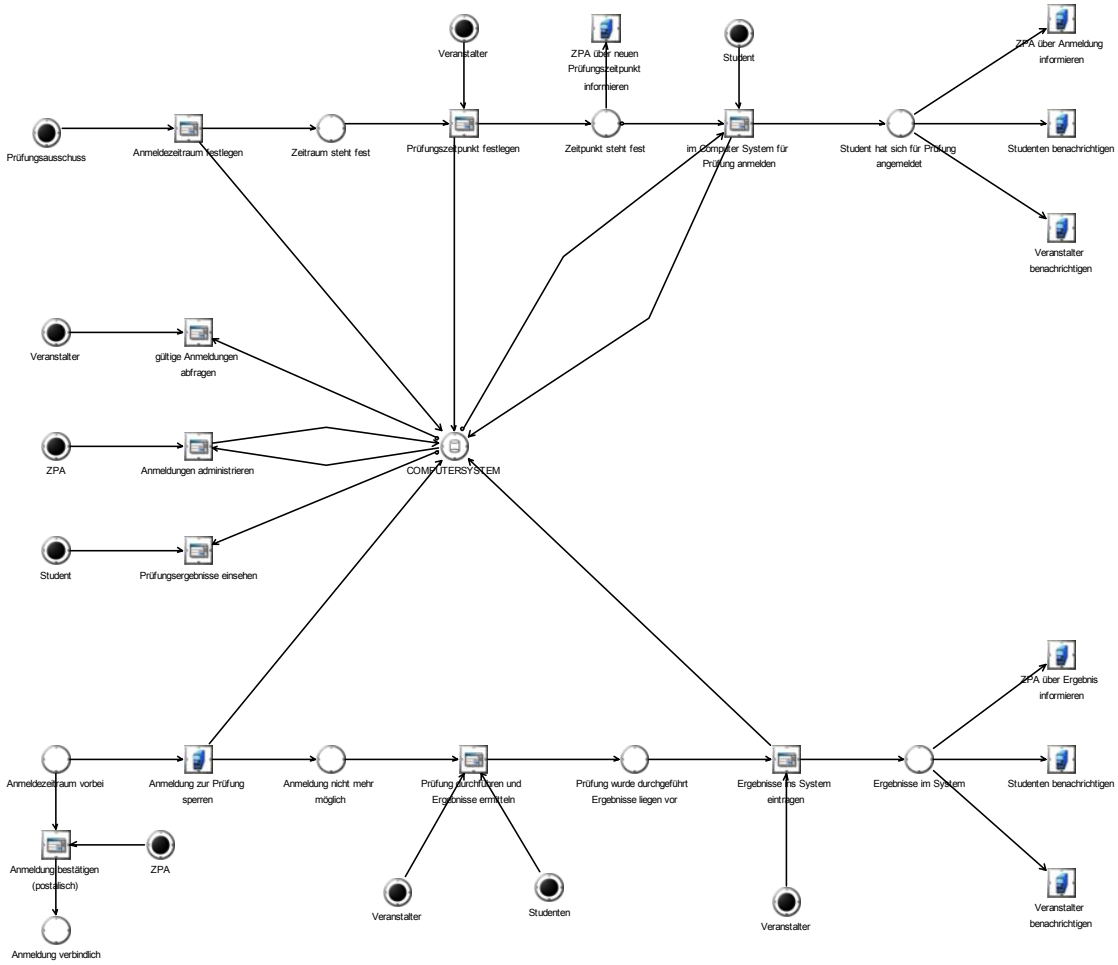


Abbildung 7-20: Anmeldung zu einer schriftlichen benoteten Prüfung

Prüfungsterminfestlegung und -anmeldung

Nachdem der Prüfungsausschuss den Prüfungs- und Anmeldezeitraum festgelegt hat, wird der Prüfungstermin vom Veranstalter festgelegt und im Computersystem eingetragen. Fortan kann der Termin von allen berechtigten Benutzern des Systems (ZPA, Veranstalter, Studenten,...) eingesehen werden. Das ZPA wird automatisch durch das System benachrichtigt, dass ein neuer Prüfungstermin im System angelegt wurde und kann auf Wunsch eine Übersicht über alle Neuanmeldungen bekommen. Der Student ist ab diesem Zeitpunkt in der Lage, sich für die Prüfung anzumelden, falls alle notwendigen Vorleistungen erfüllt sind; was vom System überprüft wird. Bei fehlenden Vorleistungen wird dem Studenten die Anmeldung verwehrt. Zur Anmeldung muss der Student keine weiteren Angaben machen, da alle bisher von Hand eingetragenen Daten wie z.B. Matrikelnummer, Studiengang, Semesterzahl, usw. direkt aus dem Computersystem entnommen werden können.

Prüfungsdurchführung und Ergebnisermittlung

Nachdem der vom Prüfungsausschuss festgelegte Anmeldezeitraum vorbei ist, werden automatisch vom Computersystem keine weiteren Anmeldungen mehr angenommen. Der Student erhält auf postalischem Weg seine Anmeldebestätigung. Damit ist der Prüfungstermin verbindlich. Nach durchgeführter Prüfung und Notenermittlung trägt der Veranstalter das Prüfungsergebnis in das System ein. Unmittelbar danach steht das Prüfungsergebnis im System für den Studenten, den Prüfer sowie das ZPA zum Abruf bereit. Alle Studenten, die an der Prüfung teilgenommen haben und deren Ergebnisse im System vorliegen, werden automatisch benachrichtigt. Die eigentliche Note kann nur nach einem Login ins Computersystem über eine sichere Verbindung erfragt werden.

Systemabfragen
Der berechtigte ZPA-Mitarbeiter hat die Möglichkeit, alle eingegangenen Neuanmeldungen einzusehen. Er soll zu jedem Zeitpunkt die Möglichkeit haben, Anmeldungen abzuweisen. Der Student kann daraufhin über das Computersystem benachrichtigt werden (z.B. per e-Mail) sowie über die bisher genutzten Kommunikationswege (Telefon, postalischer Weg).

Der Veranstalter soll zu jedem Zeitpunkt einen Überblick über alle gültigen Anmeldungen durch das Computersystem abfragen können.

Der Student soll zu jedem Zeitpunkt einen Überblick über seine Prüfungsergebnisse durch das Computersystem abfragen können.

7.2.2.3 Die schriftliche unbenotete Prüfung

Dieses Modell beschreibt den Anmeldevorgang bei schriftlichen unbenoteten Prüfungen (i.d.R. Scheinklausuren).

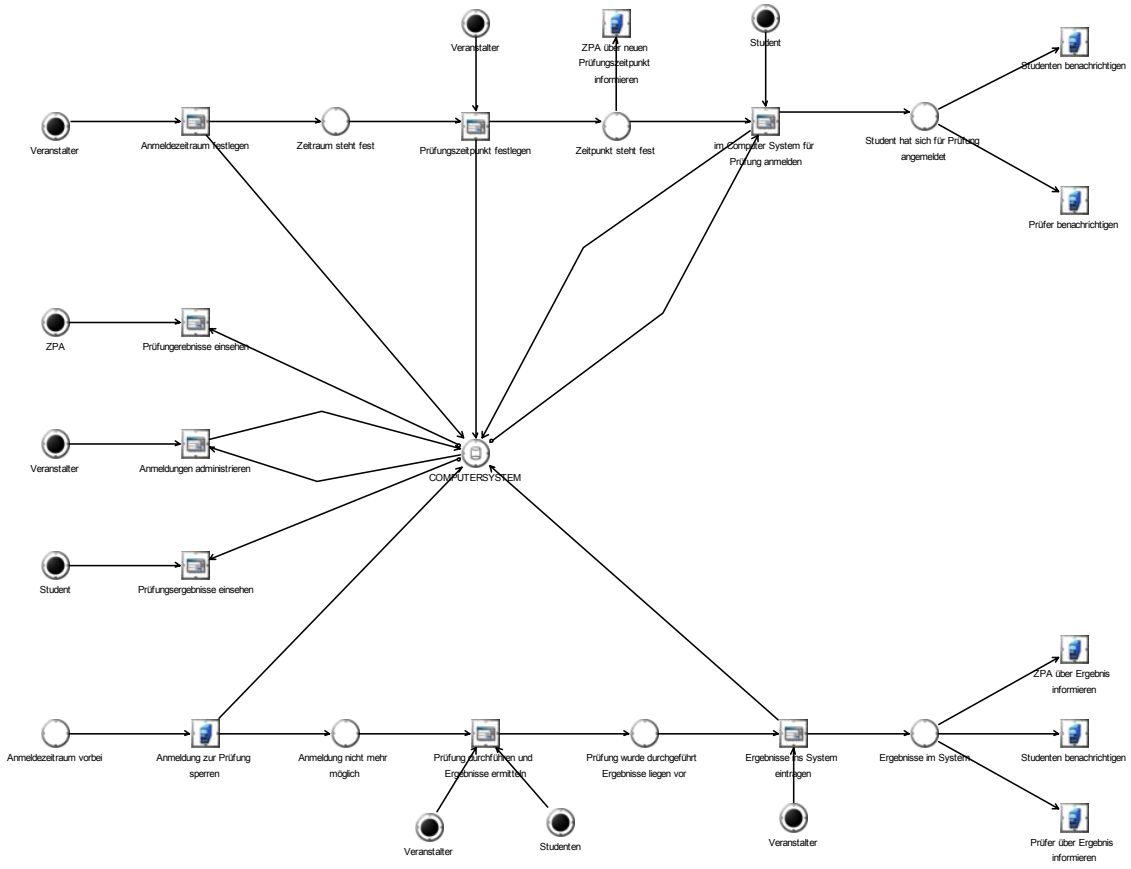


Abbildung 7-21: Anmeldevorgang bei schriftlichen unbenoteten Prüfungen (i.d.R. Scheinklausuren)

Prüfungsterminfestlegung und -anmeldung
Nachdem der Veranstalter einen Anmeldezeitraum und den Prüfungszeitpunkt im System eingetragen hat, wird das ZPA automatisch über den Prüfungstermin informiert. Der Student kann sich nun für die Prüfung anmelden. Nach erfolgter Anmeldung werden Student und Prüfer über das System benachrichtigt.

Prüfungsdurchführung und Ergebnisermittlung
Ist der Anmeldezeitraum verstrichen, werden automatisch weitere Anmeldungen vom System verhindert. Nach erfolgter Prüfung und Prüfungsermittlung werden die Ergebnisse vom Veranstalter in das System eingetragen. Daraufhin werden das ZPA, der Student und der Prüfer über das Ergebnis informiert.

Systemabfragen
Das ZPA hat jederzeit die Möglichkeit, Prüfungsergebnisse einzusehen.

Der Veranstalter hat jederzeit die Möglichkeit, Anmeldungen zu administrieren. So kann er z.B. einzelnen Studenten die Anmeldung ablehnen.

Der Student ist nach erfolgter Prüfung jederzeit in der Lage, seine Prüfungsergebnisse einzusehen.

7.2.2.4 Abmeldevorgang bei benoteten Prüfungen

Dieses Modell beschreibt den Prozess der Abmeldung bei benoteten Prüfungen.

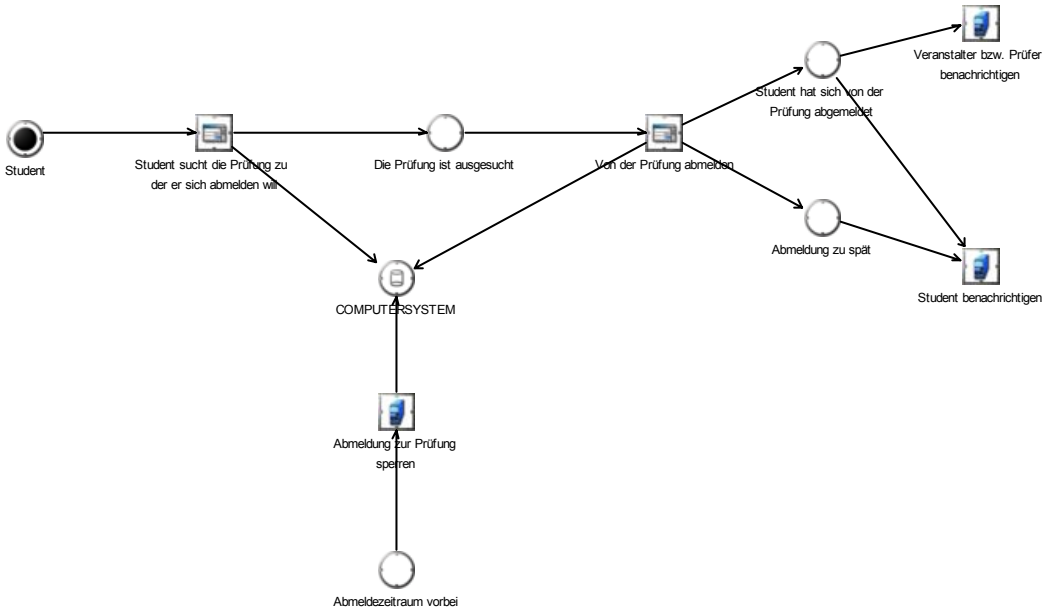


Abbildung 7-22: Abmeldung von einer benoteten Prüfungen

Prüfungsabmeldungen
Prüfungsabmeldungen lässt das System ebenfalls nach erfolgtem Login zu. Der Student sucht sich dazu in der Liste der Prüfungen, zu denen er angemeldet ist, diejenige aus, von der er sich abmelden will. Er sieht sogleich, ob der Zeitraum für die Abmeldung noch nicht abgelaufen ist und kann nach einer Bestätigung die Abmeldung veranlassen. Danach erhält der Prüfer bzw. der Veranstalter eine Meldung über die Abmeldung des Studenten. Ist der Abmeldungszeitraum schon vorüber, so kann sich der Student trotzdem abmelden, muss aber das Attest nachreichen, damit die Abmeldung rechtskräftig wird.

7.3 Grober Systementwurf

An dieser Stelle steht eine sehr stark vereinfachte Formulierung der Grundfunktionen der von Clevery unterstützen Geschäftsbereiche. In den folgenden Unterkapiteln wird auf die einzelnen Aspekte des Systems recht abstrakt eingegangen und es werden daraus zum Teil auch Forderungen an den Feinentwurf formuliert. Dabei werden hier alle Überlegungen dokumentiert, die in dieser Projektphase angestellt wurden, auch diejenigen, die später keine Berücksichtigung fanden. Die genauen Ausprägungen der realisierten Systemfunktionen sind Bestandteil des Kapitels über den Feinentwurf.

7.3.1 Allgemein

Die Zielsetzung für den Systementwurf sieht ein rollenbasiertes System mit den Hauptakteuren Administrator, Student, Mitarbeiter, Prüfer, ZPA, Veranstalter und Gast vor. Jeder Rolle, mit Ausnahme der Gastrolle, sollen vom Administrator zusätzliche Tätigkeiten erlaubt oder verboten werden können.

7.3.2 Benutzerschnittstelle

Die Benutzerschnittstelle muss allen Rollen gerecht werden. Da als Medium für diese Applikation ein Browser dient, müssen die Benutzerschnittstellen entsprechend entworfen werden. Alle Funktionen des Systems sollen über die Web-Oberfläche verfügbar sein. Als besondere Funktionalität, soll ein eingeschränkter Teil des Systems, der in erster Linie die Belange eines Studenten deckt, über WML-Seiten steuerbar sein, so dass diese Funktionen per WAP-Mobiltelefon verfügbar sind.
Grundsätzlich soll das Design einheitlich sein, abgesehen von dem WAP-Client, der ein eigenes, dem Handy angemessenes, Design erhält. Die Menüs an sich sind immer den angemeldeten Rollen angepasst, so dass kein Benutzer durch Funktionen verwirrt wird, die für ihn keine Relevanz haben. Auch soll die Oberfläche durch diese Maßnahmen schlank und transparent wirken. Wichtig für die spätere Verwendung ist eine sich weitestgehend selbsterklärende Oberfläche.

7.3.3 Übungsgruppenmanagement

Primär soll sich die Rolle Student zu einer Übungsgruppe anmelden können. Die Gast-Rolle erhält nur eine Übersicht über die aktuellen Übungsgruppenangebote ohne Anmeldungsmöglichkeit. Der Veranstalter soll aktuelle Informationen zu Übungsgruppen einstellen können und soll die Möglichkeit haben Informationen per E-Mail-Verteiler / Newsletter zu verbreiten.

7.3.3.1 Möglicher Ablauf des Übungsgruppenmanagements

1. Der Veranstalter stellt verschiedene Übungsgruppentermine mit Räumlichkeiten (max. Anzahl an Teilnehmern) und Anmeldeschluss ins System.

2. Der Benutzer / Gast erhält eine aktuelle Übersicht des Angebots.

3. Der Benutzer wählt bis zu 3 Termine aus (mit persönlicher Priorität).

4. Nach dem Anmeldeschluss stößt der Veranstalter eine automatische Verteilung an, in der die Benutzer auf die Termine verteilt werden.

5. Der Benutzer erhält eine Benachrichtigung per E-Mail, zu welcher Gruppe er zugeordnet ist. Spätere Änderungen sind durch den Veranstalter manuell möglich.

7.3.3.2 Optionen für das Übungsgruppenmanagementsystem

Die folgend aufgelisteten Punkte sind Funktionen, die bei der anstehenden Implementierung ins Auge gefasst werden. Allerdings müssen die Relevanz dieser Funktionen und deren Realisierbarkeit diskutiert werden:

- Punktevergabe
- Elektronische Abgabe von Übungsblättern
- Druckfunktionalitäten zum Ausgeben von Listen usw.
- Automatische Raumkollisionsvermeidung
- Intelligenter Verteilungsalgorithmus
- Anmeldung auch per SMS
- Beschränkung nicht nur auf Übungsgruppen, evtl. Anmeldungsportal für jede Veranstaltung, die eine Anmeldung / E-Mail-Verteiler etc. benötigt: Seminare, Praktika, außergewöhnliche Veranstaltungen (Gastvorträge mit Teilnehmerbeschränkung), private Veranstaltungen Diskussionsforen (allgemein, zu bestimmten Übungsgruppen / Veranstaltungen, etc.)

7.3.3.3 Sicherheit beim Übungsgruppenmanagement

Die Sicherheit ist bei dem gesamten System von großer Wichtigkeit. Aus diesem Grund werden hier die gleichen Mechanismen eingesetzt wie bei den sensibleren Prüfungsbereichen auch wenn dieser Bereich im Bezug auf Sicherheit eher unproblematisch ist. Gefordert werden daher an dieser Stelle:

- Authentifizierung, jeder Benutzer muss dem System bekannt sein.
- Anmeldung des Benutzers mit Name, Passwort und E-Mail-Adresse
  - Schlüssel wird generiert und dem Benutzer zugesandt.
  - Einmalige Eingabe des Schlüssels bei der ersten Anmeldung.
- Dateneingabe der Veranstalter kann lokal beschränkt werden (z.B. Uni-Netz).

7.3.4 Prüfungsmanagement

Dieser Bereich des Systems ist wesentlich sensibler als der Übungsgruppenteil. Durch die juristisch bindenden Hintergründe bei Prüfungen und der damit verbundene Begehrlichkeit im Bezug auf Datenmanipulation, ist in diesem Teil besondere Sorgfalt gefragt. Auch das Timing ist hier sehr wichtig, da durch das System verschuldete Komplikationen oder Fristverletzungen zu einem organisatorischem Mehraufwand führen können, der in keinem Verhältnis zu den durch das Clevery System eingesparten Ressourcen steht.

7.3.4.1 Authentifizierung beim Prüfungsmanagement

Aus den zuvor genannten Gründen liegt hier ein Schwerpunkt des Prüfungsteils. Die Lösungsansätze sehen wie folgt aus:

- Eine eigene Lösung mit eigener Benutzerdatenbank und evtl. Authentifizierung mit Chipkarten (digitale Unterschrift)
- Anbindung ans HRZ („E-Mail für Alle“: Benutzername / Passwort)

- Sicherung der persönlichen Daten auf die zugegriffen werden muss (Matrikel-Nr., Name, Geburtstag, Vorleistungen, bereits bestandene Prüfungen, etc.) durch spezielle Mechanismen oder Verschlüsselung

- Sicherung der sensiblen Funktionen wie z.B. der Noteneingabe durch spezielle Mechanismen oder Verschlüsselung gekoppelt mit sehr strengem Rechtemanagement für die exekutiven Rollen Prüfer und ZPA

7.3.4.2 Mögliche Funktionalitäten des Prüfungsmanagements

Die Kommunikation der beteiligten Rollen ist beim Prüfungsmanagement viel verschachtelter als beim Übungsgruppenmanagement. Daher werden die geforderten Funktionalitäten nach Rollen getrennt dargestellt. Die Kommunikation zwischen den Rollen gerät dabei zwar ins Hintertreffen, dafür sind die Funktionen für die einzelnen Rollen klarer erkennbar:

Prüferseite:

- Eigener Kalender/Terminverwaltung (Vorteil: Akzeptanz, Integration, Nachteil: Aufwand)
- Prüfer (bzw. Sekretärin o.ä.) kann Prüfungstermine veröffentlichen
- Prüfer soll Informationen zu seinen Prüfungsterminen (welche Prüfungen abgenommen werden können, welche Scheine vorausgesetzt werden usw.) veröffentlichen können
- Einsicht in die Teilnehmerliste für seine Termine
- Nach erfolgter Prüfung sollen die Prüfungsnote und das Prüfungsprotokoll eingetragen werden können
- Statistiken sollen abfragt werden können

Studentenseite:

- Leistungen, wie z.B. Praktika, sollen selbständig eingegeben werden können
- DPO soll eingesehen werden können
- Abruf eines beispielhaften Studienverlaufs
- Prüfungstermine sollen einsehbar sein
- wenn sich ein Student zur Prüfung anmeldet, soll dies an das ZPA weitergeleitet und dort verarbeitet werden

ZPA-Seite:

- Anmeldungen sollten in Sammelübersicht einsehbar sein
- Email o.a. Benachrichtigung sollen bei neuen Anmeldungen, Abmeldungen oder sonstigen Änderungen automatisch an das ZPA geleitet werden
- Export-Funktion für die HIS-Datenbank soll bereitgestellt werden
- ZPA soll Möglichkeit haben, über ein Clevery-Interface den Studenten vorformulierte Standardmitteilungen zukommen zu lassen

7.3.5 Grobarchitektur

Technisch gesehen ist der Grobentwurf des Systems eine klassische 3-Tier Architektur. Ein Browser bildet dabei das Frontend, eine Datenbank das Backend und als Middleware werden ein Web Container und ein EJB Container eingesetzt.

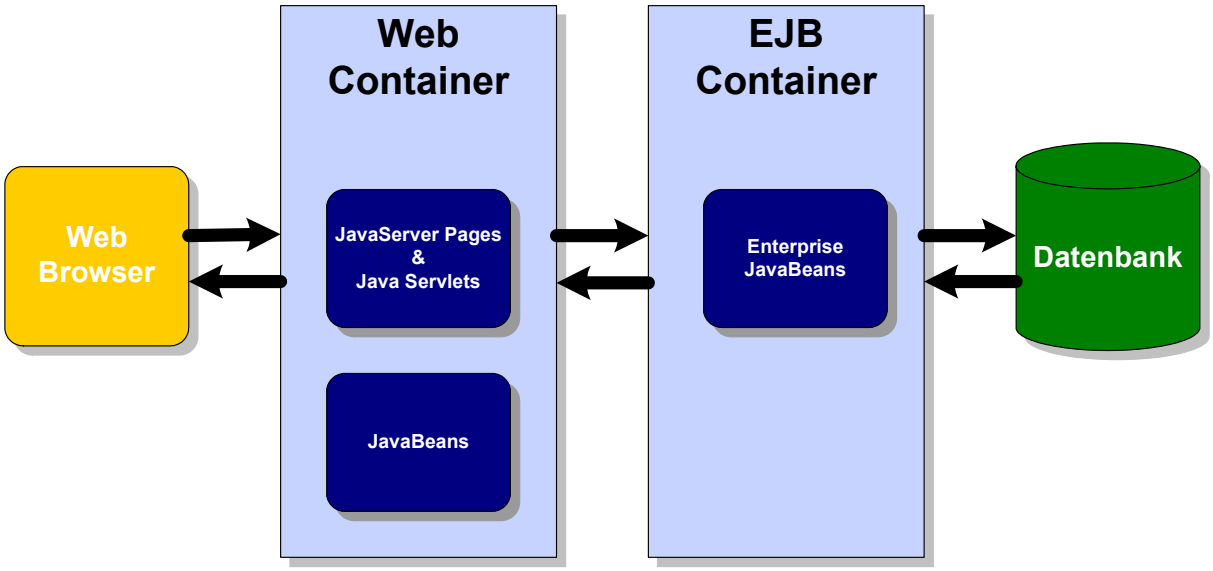


Abbildung 7-23: 3-Tier Architektur

Die konkrete Ausprägung dieser Architektur für das „Clevery“ System sieht folgendermaßen aus:

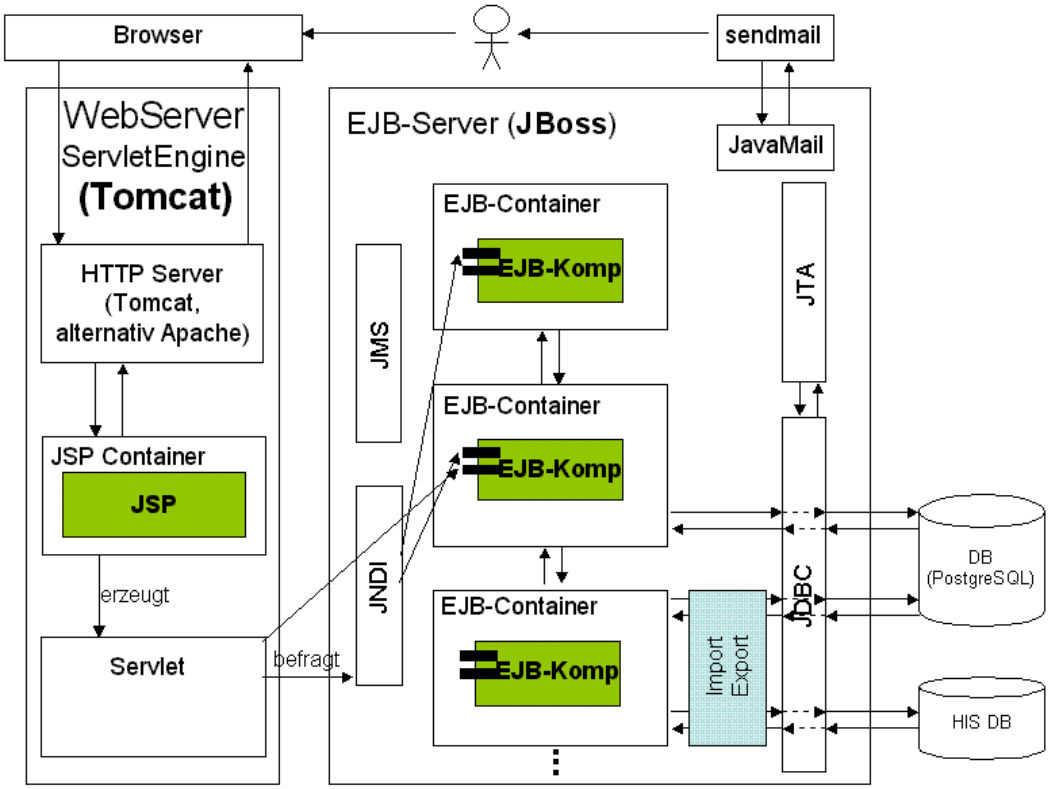


Abbildung 7-24: Grobarchitektur

Auf die einzelnen Komponenten dieser Architektur wird an dieser Stelle nicht weiter eingegangen. Eine genauere Beschreibung dieser Komponenten und den damit gesammelten Erfahrungen ist im Kapitel über die Systemimplementierung enthalten. Clevery verwendet in der endgütigen Version nur einen EJB-Container. Weitere EJB-Container für zusätzliche Aufgaben ließen sich problemlos integrieren, aber für die Umsetzung der PG-Ziele wäre eine Verteilung der Funktionen auf mehrere Container nicht sinnvoll gewesen. Zwei Aspekte dieses Entwurfs sind bei dem Projekt nicht umgesetzt worden:

1. Eine Anbindung an die HIS Datenbank ist nicht implementiert worden. Die HIS hatte sich gesträubt mit der PG zusammenzuarbeiten, so dass diese Anbindung nicht realisiert werden konnte. Über das JDBC Interface wäre eine solche Anbindung nicht schwer zu implementieren, lediglich bei der Businesslogik müssten Änderungen vorgenommen werden. Der unterste EJB-Container in der Abbildung „Grobarchitektur“ wäre in erster Linie für diesen Datenaustausch mit der HIS-Datenbank zuständig gewesen.

2. Der HTTP-Server wird für das System nicht verwendet. Alle Systemseiten werden dynamisch erstellt. Statischen Content gibt es in Clevery nicht und somit ist der HTTP-Server überflüssig geworden. Ein zusätzliches Angebot an statischen Webseiten könnte aber, sofern gefordert, eben über einen solchen HTTP-Server zur Verfügung gestellt werden. Der „Apache“ Webserver böte sich in einem solchen Fall an, da es sich bei ihm, wie beim „Tomcat“ Container, um ein freies Produkt handelt und beide Applikationen oft als Kombination eingesetzt werden.

7.4 Feinentwurf

Dieser Abschnitt beschreibt die Entwicklung des Projektes der PG nightshift aus technischer Sichtweise. Der Schwerpunkt liegt dabei auf dem Kapitel Szenarios. Die entsprechenden Klassen- und Interaktionsdiagramme sind nur als Together-Projekt gespeichert. Ein Ausdruck dieser Diagramme ist aufgrund der sehr komplizierten Hierarchie nicht sinnvoll. Die Interaktion der Rollen wird daher in dem Abschnitt „Szenarios“ verbal verdeutlicht. Weiterhin werden auch die von der PG benutzten Methoden zu der Entwicklung der Klassen- und Interaktionsdiagramme beschrieben.

7.4.1 Design Patterns

An dieser Stelle werden zwei der von uns benutzten Design Patterns beschrieben. Als Grundlage diente eine Ausarbeitung der Gang of Four [DP] die dann an unsere speziellen Bedürfnisse bzgl. der Enterprise JavaBeans Technologie angepasst wurde. Diese sind in unserer PG unter dem in diesem Dokument benutzten Begriff „Gang of Eleven Design Patterns“ bekannt.

7.4.1.1 „Value Object“

Dies entspricht einem Design Pattern der Gang of Four. Dabei erfolgt der Zugriff auf Daten nicht einzeln für jedes Attribut, sondern über ein sog. Value Object, welches eine Sammlung der gewünschten Daten darstellt. Dadurch wird der Netzwerkverkehr deutlich entlastet.

7.4.1.2 „Session Facade“

Das Session Facade Pattern sieht vor, alle Geschäftsprozesse, die auf Geschäftsobjekte zugreifen in einer SessionBean zu kapseln, damit nicht direkt auf die EntityBeans zugegriffen wird. Hierdurch wird vor allem die Transaktionssicherheit gewährleistet. Das Pattern dient aber auch der Sicherheit, da man einem Geschäftsprozess detaillierter Rechte zuweisen kann als einem Geschäftsobjekt.

7.4.1.3 Beispiele

Schreibzugriff: In `UebVeranstalterController.updateUebung (UebungDetail details)` wird die Entity Uebung (über die Felder des ValueObjects UebungDetail) geändert, jedoch nur die Felder, auf die die Rolle UebVeranstalter (ÜV) Zugriff hat. Die restlichen Felder bleiben unberührt.

Lesezugriff: Die finder-Methoden liefern eine Collection von ValueObjekten in denen nur die Feldern belegt sind, die die Rolle lesen darf. `public PruefungLeistungDetailStudentControllerBean.findPruefungsLeistung(StudentID string, PruefungID string)` liefert ein ValueObject in dem nur die Felder der PruefungsLeistung belegt sind, die der Student sehen darf.

7.4.2 Szenarios

Die Szenarios dienen als Vorlage für die Abläufe der einzelnen Funktionen des Systems. Ein Szenario beinhaltet somit die beteiligten Rollen, die Voraussetzungen, die erfüllt sein müssen damit die Funktion ausgeführt werden kann und den sequentiellen Ablauf der beschriebenen Funktion. Sie zeigen zusätzlich die Kommunikationsstruktur des Systems auf und ersetzen damit quasi Interaktionsdiagramme.

7.4.2.1 Aufbau

Die Szenarios haben alle den gleichen Aufbau:

- Zuerst werden die Voraussetzungen aufgeführt, die erfüllt sein müssen, damit die in dem Szenario dargestellte Aktion ausgeführt werden kann. Die Voraussetzungen werden als Aufzählung dargestellt, weil es für den weiteren Ablauf unerheblich ist, in welcher Reihenfolge die Voraussetzungen erfüllt wurden.
- Als zweites werden die einzelnen Schritte dargestellt, die gemacht werden müssen bis die Aktion durchgeführt ist. Diese ist ein sequentieller Ablauf und daher durch eine Aufzählung dargestellt.

Das Layout der Szenarios hat somit folgende Form:

Voraussetzungen:

- ...

Ablauf:

1. ....
2. ...

7.4.2.2 Beteiligte Rollen

Für die im Folgenden beschriebenen Szenarios ist eine Übersicht aller beteiligten Rollen wichtig. Die hier aufgeführten Rollen werden dem Benutzer beim Login zugeordnet. Im System werden die Rollen eines Benutzers als Attribute der Entität „Benutzer“ festgehalten. Die Vergabe der Rollen findet durch Benutzer mit entsprechenden Rechten statt, z.B. kann der Administrator einem Benutzer die Rolle Übungsgruppenveranstalter zuweisen, so dass sich dieser Benutzer dann nach dem Login in genau dieser Rolle befindet und die entsprechenden Befugnisse hat. Die folgenden Rollen werden in Clevery benutzt:

Rolle	Titel	Beschreibung und Zugriffsrechte
BEN	Benutzer	Bezeichnet den Benutzer des Systems, wie er in der realen Welt existiert, d.h. mit Name, Adresse,..., Benutzername und Passwort. Er kann. • sein Profil editieren.
GAST	Gast	Benutzer mit nur sehr eingeschränkten Rechten, die allerdings auch ohne Login zur Verfügung stehen. Er kann. • nur öffentliche Daten einsehen.
STD	Student	Typischer Benutzer des Systems. Für ihn sollen An- und Abmeldungen zu Übungen und Prüfungen möglich sein und er soll eine Übersicht über bereits

Rolle	Titel	Beschreibung und Zugriffsrechte
		erbrachte Leistungen erhalten können. Er kann: • sich zu Übungen und Prüfungen anmelden • lesend auf seinen Übungsstatus zugreifen
ÜV	Übungsveranstalter	in älteren Dokumenten auch als Vorlesungsveranstalter bezeichnet. Er kann: • Übungsveranstaltungen anlegen und editieren sowie seine eigenen löschen • den ÜO für seine Übungsveranstaltungen festlegen
ÜGL	Übungsgruppenleiter	Leiter von einer oder mehrerer Übungsgruppen. Dieser kann ein wissenschaftlicher Angestellter der Universität oder eine studentische Hilfskraft sein. Er kann: • alle Übungsgruppen seiner Übung(en) verwalten
ÜO	Übungsorganisator	Zentrale Rolle im Übungsbetrieb. Er kann: • Übungsgruppen anlegen, editieren und löschen • alle Übungsgruppen verwalten
ADMIN	Administrator	Zentrale Rolle des gesamten Systems. Er kann: • alle technischen und inhaltlichen Parameter des Systems modifizieren
PRF	Prüfer	Prüfer, meist ein Professor. Er kann • schriftl. sowie mündl. Prüfungen anlegen • Noten dazu eintragen • An- und Abmeldungen an- und ablehnen
VER	Veranstalter	Metarolle, die für die Abbildung der Geschäftsprozesse im System vorkommt, aber keine separaten Rechte besitzt.
PA	Prüfungsausschuss	Rolle, die für die Abbildung der Geschäftsprozesse im System vorkommt, aber keine separaten Rechte besitzt. Clevery stellt für den PA keine Funktionen bereit.
ZPA	Zentrales Prüfungsamt	Zentrale Rolle im Prüfungsmanagement. Es kann: • Prüfungsanmeldungen und -abmeldungen annehmen und ablehnen • Noten einsehen
TN	Teilnehmer	Eine Subrolle, die sich durch das Vorhandensein entsprechender Relationen ergibt. Subrollen werden durch den Login zugewiesen und sind daher keine Attribute der Entität Benutzer
SYS	das System	Diese Rolle wird zu Modellierung und Beschreibung der Szenarios benutzt, ist aber keine eigenständige Rolle.
UPDB	Universelle Projektdatenbank	Auch diese Rolle tritt nur als Abkürzung im weiteren Text auf und hat keine eigenständige Funktionalität. Sie beschreibt nur den Datenspeicher des Systems.

7.4.2.3 Übersicht über die erstellten Szenarios

Allgemein

- Anmelden am System

Übungsbetrieb

Rolle Gast

- Vorlesungsübersicht anzeigen

Rolle Student
- Anmeldung zu einer Übungsgruppe

Rolle Übungsveranstalter
- Übungsveranstaltung anlegen
- Übungsveranstaltung ändern
- Übungsveranstaltung löschen

Rolle Übungsgruppenleiter
- Anwesenheitsliste eingeben
- Punkte verteilen/eingeben
- Nachträgliche Eintragung eines neuen Teilnehmers
- Nachträgliche Verschiebung eines Teilnehmers
- Nachträgliche Änderung der Daten eines Teilnehmers
- Nachträgliche Löschung eines Teilnehmers (aus einer Übungsgruppe)
- Ausdrucken der Scheine

Rolle Übungsorganisator
- Anmeldephase einstellen
- Übungsgruppe anlegen
- Übungsgruppe ändern
- Übungsgruppe löschen
- Verteilung der Studenten auf die Übungsgruppen
- Termine bekannt geben

Rolle Administrator
- Verteilung der Studenten auf die Übungsgruppen

Prüfungsbetrieb

Mündliche Prüfung
- Terminabsprache
- Terminreservierung
- Terminbestätigung
- Anmeldung beim ZPA
- Prüfungsdurchführung
- Prüfungsnachbearbeitung
- Prüfer- und Fächerkombination im System anlegen
- Prüfungsabmeldung
- Abmeldebearbeitung

Schriftliche unbenotete Prüfung
- Festlegen des Prüfungstages
- Festlegen des Anmeldezeitraums
- Anmeldung des Studenten
- Bearbeitung durch den Veranstalter
- Noteneintragung
- Prüfungsnachbearbeitung
- Note abfragen

Schriftliche benotete Prüfung
- Festlegen der Prüfungs- und Anmeldezeitraums
- Festlegen des Prüfungstages
- Anmeldung des Studenten
- Bearbeitung durch das ZPA
- Prüfungsdurchführung
- Prüfungsnachbearbeitung

Benutzerinformationen
- Statusabfragen des Studenten
- Statusabfrage des Veranstalters

7.4.3 Allgemein

Die Funktion der Systemanmeldung ist sowohl für den Übungsbetrieb als auch für den Prüfungsbetrieb erforderlich, hat also allgemeinen Charakter. Um Wiederholungen zu vermeiden, wird dieses Szenario an dieser Stelle separat dargestellt.

7.4.3.1 Anmelden am System

Voraussetzungen:
• keine
Ablauf:
3. BEN gibt Benutzerkennung ein 4. BEN gibt Benutzerpasswort ein 5. SYS überprüft die Zugangsdaten 6. SYS ermittelt/setzt die Zugriffsrechte 7. SYS bestätigt die Anmeldung 8. SYS teilt die aktuelle Rolle dem BEN mit 9. SYS zeigt verfügbare Funktionen für den BEN an

7.4.4 Übungsbetrieb

Die Szenarios für den „Übungsbetrieb“ sind nach den darin vorkommenden Rollen, z.B. „Gast“, „Student“ usw., gruppiert. Diese Gruppierung ist sinnvoll, da sich die Aktionen der einzelnen beteiligten Rollen nicht in einem Kausalgefüge befinden, d.h. die Aktion einer Rolle ist nicht abhängig von einer früher abgeschlossenen Aktion einer anderen Rolle. Hier werden alle den jeweiligen Rollen zugeordneten Funktionen dargestellt, die den Übungsbetrieb unterstützen.

7.4.4.1 Rolle Gast

7.4.4.1.1 Vorlesungsübersicht anzeigen

Voraussetzungen:
• keine
Ablauf:
10. GAST fordert Vorlesungsübersicht an 11. SYS zeigt die verfügbaren Vorlesungen

7.4.4.2 Rolle Student

7.4.4.2.1 Anmeldung zu einer Übungsgruppe

Voraussetzungen:
• STD ist am System angemeldet • Zugehörige Übungsveranstaltung ist bereits eingetragen worden. • Übungsgruppen wurden eingetragen. • Übung wurde zur Anmeldung freigeschaltet
Ablauf:

12. SYS bietet Vorlesungsübersicht
13. STD sucht sich aus der Vorlesungsübersicht die gewünschte Vorlesung
14. SYS bietet Übersicht über vorhandene Übungsgruppen
15. STD sucht sich passende Termine aus
16. STD schickt Anmeldung ab
17. SYS bestätigt Anmeldung

7.4.4.3 Rolle Übungsveranstalter

7.4.4.3.1 Übungsveranstaltung anlegen

Voraussetzungen:

- ÜV ist am System angemeldet

Ablauf:

18. ÜV trägt folgende Daten ein: Titel, Beschreibung, geschätzte Teilnehmerzahl, Scheinkriterien, nötige Prioritäten, ÜO
19. SYS überprüft die Eingabe
20. SYS trägt Daten ein
21. SYS bestätigt die Eintragung

7.4.4.3.2 Übungsveranstaltung ändern

Voraussetzungen:

- ÜV ist am System angemeldet
- Vorlesung ist eingetragen

Ablauf:

22. SYS zeigt dem ÜV alle Übungsveranstaltungen an, die geändert werden können
23. ÜV wählt zu ändernde Übung aus
24. SYS zeigt alle Daten zu dieser Übung an
25. ÜV ändert Daten
26. SYS bestätigt Eingabe

7.4.4.3.3 Übungsveranstaltung löschen

Voraussetzungen:

- ÜV ist am System angemeldet
- Vorlesung ist eingetragen

Ablauf:

27. SYS zeigt dem ÜV alle Übungsveranstaltungen an, und markiert die, die gelöscht werden können
28. ÜV wählt zu löschende Vorlesung aus
29. SYS erfragt Bestätigung
30. ÜV bestätigt Löschungsauftrag
31. SYS bestätigt Löschung

7.4.4.4 Rolle Übungsgruppenleiter

7.4.4.4.1 Anwesenheitsliste eingeben

Voraussetzungen:

- Benutzer ist am System angemeldet
- Zugehörige Vorlesung ist bereits eingetragen worden
- Zugehörige Übung ist bereits eingetragen worden
- Übungsgruppen wurden eingetragen
- Verteilung ist abgeschlossen

Ablauf:

32. SYS zeigt die Auswahl von Übungsgruppen für die der ÜGL registriert ist
33. ÜGL wählt eine Übungsgruppe aus
34. SYS zeigt ein Formular mit der Liste der TN
35. ÜGL wählt TN aus, die zur der Übung nicht erschienen sind und bestätigt die Auswahl
36. SYS aktualisiert die interne Liste und bestätigt die Änderungen und zeigt aktuelle Anwesenheitsstatistik tabellarisch an

7.4.4.4.2 Punkte verteilen/eingeben

Voraussetzungen:

- Benutzer ist am System angemeldet
- Zugehörige Vorlesung ist bereits eingetragen worden
- Zugehörige Übung ist bereits eingetragen worden
- Übungsgruppen wurden eingetragen
- Verteilung ist abgeschlossen

Ablauf:

37. SYS zeigt die Auswahl von Übungsgruppen für die der ÜGL registriert ist
38. ÜGL wählt eine Übungsgruppe aus
39. SYS zeigt ein Formular mit der Liste der TN Eingabefeldern für Punkte (vorinitialisiert mit 0)
40. ÜGL gibt die erreichte Punktzahl des letzten Aufgabe ein und sendet das Formular ab
41. SYS meldet die Änderung und zeigt eine tabellarische Punktestatistik an

7.4.4.4.3 Nachträgliche Eintragung eines neuen Teilnehmers

Voraussetzungen:

- ÜGL ist am System angemeldet
- Zugehörige Vorlesung ist bereits eingetragen worden
- Zugehörige Übung ist bereits eingetragen worden
- Übungsgruppen wurden eingetragen
- Verteilung ist abgeschlossen
- STD ist im SYS registriert

Ablauf:

42. SYS zeigt Suchmaske an
43. ÜGL lässt STD suchen
44. SYS zeigt alle passenden STD an
45. ÜGL wählt passenden STD aus
46. ÜGL teilt Teilnehmer einer Übungsgruppe zu (SYS bietet nur Übungsgruppen an, die modifiziert werden können)

7.4.4.4.4 Nachträgliche Verschiebung eines Teilnehmers

Voraussetzungen:

- ÜGL ist am System angemeldet
- Zugehörige Vorlesung ist bereits eingetragen worden
- Zugehörige Übung ist bereits eingetragen worden
- Übungsgruppen wurden eingetragen
- Verteilung ist abgeschlossen

Ablauf:

1. SYS zeigt Teilnehmerliste der zugehörigen Übung
2. ÜGL wählt Teilnehmer
3. ÜGL teilt Teilnehmer einer Übungsgruppe zu (SYS bietet nur Übungsgruppen an, die modifiziert werden können)

7.4.4.4.5 Nachträgliche Änderung der Daten eines Teilnehmers

Voraussetzungen:

- ÜGL ist am System angemeldet
- Zugehörige Vorlesung ist bereits eingetragen worden
- Zugehörige Übung ist bereits eingetragen worden
- Übungsgruppen wurden eingetragen
- Verteilung ist abgeschlossen

Ablauf:

1. SYS zeigt Teilnehmerliste der zugehörigen Übung
2. ÜGL wählt TN dessen Daten geändert werden sollen
3. SYS zeigt die Daten über Übungsleistung des TN
4. ÜGL ändert die betreffenden Daten und bestätigt die Änderungen
5. SYS ändert Daten im System und bestätigt den Vorgang

7.4.4.4.6 Nachträgliche Löschung eines Teilnehmers (aus einer Übungsgruppe)

Voraussetzungen:

- ÜGL ist am System angemeldet
- Zugehörige Vorlesung ist bereits eingetragen worden
- Zugehörige Übung ist bereits eingetragen worden
- Übungsgruppen wurden eingetragen
- Verteilung ist abgeschlossen

Ablauf:

1. SYS zeigt Teilnehmerliste der zugehörigen Übung
2. ÜGL wählt Teilnehmer zum Löschen aus
3. SYS entfernt den Teilnehmer aus der Übungsgruppe und bestätigt den Vorgang

7.4.4.4.7 Ausdrucken der Scheine

Voraussetzungen:

- ÜGL ist am System angemeldet
- Zugehörige Vorlesung ist bereits eingetragen worden
- Zugehörige Übung ist bereits eingetragen worden

Ablauf:

1. SYS zeigt Liste aller Übungen
2. ÜGL wählt Übung aus
3. SYS zeigt Liste der TN die Scheinkriterien erfüllen
4. ÜGL wählt einzelne oder alle Studenten
5. SYS generiert gewünschte Ausgabeliste

7.4.4.5 Rolle Übungsorganisator

7.4.4.5.1 Anmeldephase einstellen

Voraussetzungen:

- ÜO ist am System angemeldet
- Zugehörige Übungsveranstaltung ist bereits eingetragen worden
- Übungsgruppen wurden eingetragen

Ablauf:

1. ÜO stellt Flag für Anmeldephase ein
2. SYS prüft ob Anzahl der anzugebenden Prioritäten die Zahl der Übungsgruppen nicht übersteigt
3. SYS bestätigt die Freischaltung

7.4.4.5.2 Übungsgruppe anlegen

Voraussetzungen:

- ÜO ist am System angemeldet
- Zugehörige Übungsveranstaltung ist bereits eingetragen worden

Ablauf:

1. ÜO trägt Räume und Termine ein
2. ÜO trägt die ÜGL ein
3. SYS trägt die Daten ein
4. SYS bestätigt die Eintragung

7.4.4.5.3 Übungsgruppe ändern

Voraussetzungen:

- ÜO ist am System angemeldet
- Zugehörige Übungsveranstaltung ist bereits eingetragen worden
- Zugehörige Übungsgruppe ist eingetragen

Ablauf:

1. ÜO wählt die Übung aus
2. SYS präsentiert Liste aller zugehörigen Übungsgruppen
3. ÜO wählt zu ändernde Übungsgruppe aus
4. SYS präsentiert alle Verwaltungsdaten der Übungsgruppe
5. ÜO führt Änderungen durch
6. SYS bestätigt Änderungen

7.4.4.5.4 Übungsgruppe löschen

Voraussetzungen:

- ÜO ist am System angemeldet
- Zugehörige Übungsveranstaltung ist bereits eingetragen worden
- Zugehörige Übungsgruppe ist eingetragen

Ablauf:

1. ÜO wählt die Übung aus
2. SYS präsentiert Liste aller zugehörigen Übungsgruppen
3. ÜO wählt zu löschende Übungsgruppe aus
4. SYS erfragt Bestätigung
5. ÜO bestätigt Löschungsauftrag
6. SYS bestätigt Löschung

7.4.4.5.5 Verteilung der Studenten auf die Übungsgruppen

Voraussetzungen:

- ÜO ist am System angemeldet
- Zugehörige Vorlesung ist bereits eingetragen worden
- Zugehörige Übung ist bereits eingetragen worden
- Übungsgruppen wurden eingetragen
- Übungsgruppe wurde freigeschaltet
- Anmeldephase wurde beendet

Ablauf:

1. SYS zeigt die verfügbaren Vorlesungen/Übungen an
2. ÜO wählt Vorlesung/Übung aus
3. SYS fordert zur Bestätigung zum Anstoß der Verteilung auf
4. ÜO stößt Verteilung der Studenten auf die Übungsgruppen an
5. SYS verteilt die Studenten bestmöglich und gibt Liste von nicht zuteilbaren Studenten zurück
6. ÜO muss die verbleibenden Studenten manuell zuteilen
7. SYS zeigt Übersicht über die Übungsgruppenbelegung

7.4.4.5.6 Termine bekannt geben

Voraussetzungen:
• ÜO ist am System angemeldet • Zugehörige Vorlesung ist bereits eingetragen worden • Zugehörige Übung ist bereits eingetragen worden • Übungsgruppen wurden eingetragen • Übungsgruppe wurde freigeschaltet • Anmeldephase wurde beendet • Verteilung ist abgeschlossen
Ablauf:
1. SYS zeigt die verfügbaren Vorlesungen/Übungen an 2. ÜO wählt Vorlesung/Übung aus 3. SYS fordert zur Bestätigung der Veröffentlichung der Termine auf 4. ÜO bestätigt die Veröffentlichung 5. SYS bestätigt die Veröffentlichung und sendet die Termine per Email zum TN

7.4.4.6 Rolle Administrator

7.4.4.6.1 Verteilung der Studenten auf die Übungsgruppen

Voraussetzungen:
• ÜO ist am System angemeldet • Zugehörige Vorlesung ist bereits eingetragen worden • Zugehörige Übung ist bereits eingetragen worden • Übungsgruppen wurden eingetragen • Übungsgruppe wurde freigeschaltet • Anmeldephase wurde beendet
Ablauf:
1. SYS zeigt die verfügbaren Vorlesungen/Übungen an 2. ÜO wählt Vorlesung/Übung aus 3. SYS fordert zur Bestätigung zum Anstoß der Verteilung auf 4. ÜO stößt Verteilung der Studenten auf die Übungsgruppen an 5. SYS verteilt die Studenten bestmöglich und gibt Liste von nicht zuteilbaren Studenten zurück 6. ÜO muss die verbleibenden Studenten manuell zuteilen 7. SYS zeigt Übersicht über die Übungsgruppenbelegung

7.4.5 Prüfungsbetrieb

Die Szenarios für den „Prüfungsbetrieb“ sind nach den Prozessen „Mündliche Prüfung“, „Schriftlich unbenotete Prüfung“ und „Schriftlich benotete Prüfung“ gruppiert. Innerhalb dieser Gruppierungen werden die Prozesse in einzelne chronologische Schritte gegliedert, die dann beschrieben werden.
An einem Prozess kann mehr als nur eine Rolle beteiligt sein und diese Rollen können in einem Kausalgefüge stehen, d.h. sie sind bei der Durchführung ihrer Aufgaben von Aktionen abhängig, die andere Rollen zuvor abgeschlossen haben müssen. Das legt die gewählte Gliederungsform nahe. Auch die Interaktionen zwischen den Rollen, dem System und auch zwischen dem System und den Rollen wird so veranschaulicht.

7.4.5.1 Mündliche Prüfung

7.4.5.1.1 Terminabsprache

Voraussetzungen:
• PRF ist am System angemeldet
Ablauf:
1. PRF ruft seinen Online-Terminplaner auf 2. PRF trägt freie Prüfungstermine ein

7.4.5.1.2 Terminreservierung

Voraussetzungen:
• STD ist am System angemeldet
Ablauf:
1. STD ruft den Online-Terminplaner des PRF auf 2. STD wählt einen freien Termin aus 3. Bemerkung: 4. Für Prüfungen, bei denen ein Zweitprüfer benötig wird muss der STD den obigen Prozess ein weiteres Mal für den Zweitprüfer durchlaufen

7.4.5.1.3 Terminbestätigung

Voraussetzungen:
• PRF ist am System angemeldet
Ablauf:
1. PRF und zweiter PRF bestätigen den Prüfungstermin (oder lehnen ihn ab) 2. SYS benachrichtigt STD entsprechend

7.4.5.1.4 Anmeldung beim ZPA

Voraussetzungen:
• Termin ist abgesprochen
Ablauf:
1. SYS informiert ZPA über Prüfungstermin 2. ZPA bestätigt Anmeldung 3. SYS benachrichtigt PRF, zweiten PRF, STD

7.4.5.1.5 Prüfungsdurchführung

Voraussetzungen:
• der STD ist angemeldet • PRF ist am SYS angemeldet
Ablauf:

4. PRF trägt Note und Protokoll ins SYS ein
5. SYS schickt die Note und das Protokoll an das ZPA

7.4.5.1.6 Prüfungsnachbearbeitung

Voraussetzungen:

- die Note und das Protokoll sind beim ZPA eingetroffen

Ablauf:

1. ZPA importiert die Note und das Protokoll in die HIS Datenbank

7.4.5.1.7 Prüfer- und Fächerkombination im System anlegen

Voraussetzungen:

- PRF ist am SYS angemeldet

Ablauf:

1. PRF trägt im SYS ein welche Prüfungen (z.B. Lindemann: RVS,...) er abnimmt

7.4.5.1.8 Prüfungsabmeldung

Voraussetzungen:

- der STD ist am SYS angemeldet
- der STD ist für die Prüfung angemeldet

Ablauf:

1. STD füllt das Formular aus
2. SYS schickt das Formular an ZPA

7.4.5.1.9 Abmeldebearbeitung

Voraussetzungen:

- Die Abmeldeformular liegt beim ZPA vor

Ablauf:

1. ZPA bestätigt die Abmeldung
2. ZPA benachrichtigt STD,PRF bzw. zweiter PRF/VER

7.4.5.2 Schriftliche unbenotete Prüfung

7.4.5.2.1 Festlegen des Prüfungstages

Voraussetzungen:

- VER ist am SYS angemeldet
- VER hat bereits einen Raum für den angedachten Prüfungstag reserviert

Ablauf:

1. VER legt den Prüfungstag fest

7.4.5.2.2 Festlegen des Anmeldezeitraums

Voraussetzungen:

- VER ist am SYS angemeldet
- VER hat den Prüfungstag festgelegt

Ablauf:

1. VER trägt den Anmeldezeitraum ins SYS ein

7.4.5.2.3 Anmeldung des Studenten

Voraussetzungen:

- STD ist am SYS angemeldet
- Prüfung ist im SYS angelegt

Ablauf:

1. STD wählt Prüfung aus
2. STD meldet sich zur Prüfung an

7.4.5.2.4 Bearbeitung durch den Veranstalter

Voraussetzungen:

- VER ist am System angemeldet
- STD hat sich zu einer Prüfung angemeldet

Ablauf:

1. VER überprüft Zulassung zur Prüfung (selten bis überhaupt nicht notwendig)
2. VER bestätigt/verweigert Prüfungsanmeldung
3. STD wird benachrichtigt

7.4.5.2.5 Noteneintragung

Voraussetzungen:

- STD ist zur Prüfung angemeldet und VER hat bestätigt
- Prüfung ist durchgeführt und Note wurde ermittelt
- VER ist am SYS angemeldet

Ablauf:

1. VER trägt Note ins SYS ein
2. SYS übermittelt Note an das ZPA

7.4.5.2.6 Prüfungsnachbearbeitung

Voraussetzungen:

- Note ist beim ZPA vorhanden
- ZPA ist am SYS angemeldet

Ablauf:

1. ZPA importiert die Note in die HIS Datenbank
2. STD wird über seine Note benachrichtigt

7.4.5.2.7 Note abfragen

Voraussetzungen:

- STD ist am SYS angemeldet
- Ergebnis liegt im SYS vor

Ablauf:

1. STD ruft Übersichtsseite mit seinen Prüfungsergebnissen auf

7.4.5.3 Schriftliche benotete Prüfung

7.4.5.3.1 Festlegen der Prüfungs- und Anmeldezeitraums

Voraussetzungen:

- PA ist am SYS angemeldet

Ablauf:

1. PA legt den Anmeldezeitraum fest
2. PA Prüfungszeitraum fest

7.4.5.3.2 Festlegen des Prüfungstages

Voraussetzungen:

- VER ist am SYS angemeldet
- Prüfungs- und Anmeldezeitraum sind festgelegt

Ablauf:

1. VER legt Prüfungstermin fest

7.4.5.3.3 Anmeldung des Studenten

Voraussetzungen:

- STD ist am SYS angemeldet

Ablauf:

1. STD wählt Prüfung aus

47. STD meldet sich zur Prüfung an

7.4.5.3.4 Bearbeitung durch das ZPA

Voraussetzungen:

- ZPA ist am System angemeldet
- STD hat sich zu einer Prüfung angemeldet

Ablauf:

1. ZPA überprüft Zulassung zur Prüfung
2. ZPA bestätigt/verweigert Prüfungsanmeldung
3. STD wird benachrichtigt

7.4.5.3.5 Prüfungsdurchführung

Voraussetzungen:

- STD ist zur Prüfung angemeldet und ZPA hat bestätigt
- PRF ist am SYS angemeldet

Ablauf:

1. PRF trägt Note ins SYS ein
2. Note und Notizen werden an das ZPA geleitet

7.4.5.3.6 Prüfungsnachbearbeitung

Voraussetzungen:
• Note und Notizen sind beim ZPA vorhanden • ZPA ist am SYS angemeldet
Ablauf:
1. ZPA importiert die Note und die Notizen in die HIS Datenbank 2. STD wird über seine Note benachrichtigt

7.4.5.4 Benutzerinformationen

7.4.5.4.1 Statusabfragen des Studenten

Voraussetzungen:
• STD ist am SYS angemeldet
Ablauf:
1. STD kann sich über erbrachte Leistungen informieren

7.4.5.4.2 Statusabfrage des Veranstalters

Voraussetzungen:
• VER am SYS angemeldet
Ablauf:
1. Der VER hat über das SYS den Zugriff auf Informationen über den STD, die für ihn relevant sind

8 Systemimplementierung

8.1 GUI Entwicklung

8.1.1 Story Board

8.1.1.1 Beschreibung

Schon früh bei der Entwicklung, als die ersten Visionen vorgestellt und besprochen wurden, traten einige Probleme zu Tage. Es zeigte sich, dass die verschiedenen Entwickler jeweils unterschiedliche Vorstellungen über einige Details hatten. Um dieser Problematik zu begegnen haben wir uns auf die Verwendung eines Storyboards geeinigt.
Ein Storyboard ist ein leicht zu modifizierender Prototyp der Benutzerschnittstelle. Dieser Prototyp spiegelt in groben Zügen das spätere Erscheinungsbild der Benutzerschnittstelle wieder. Beim Erstellen des Storyboards spielt weniger das Layout als die Funktionalität der Benutzerführung eine entscheidende Rolle. Es sollten alle Links vorhanden sein, damit der genaue Ablauf der Benutzerführung deutlich wird. Weiterhin soll ein Storyboard auch dazu dienen, die geplanten Prozesse noch einmal bis ins Detail zu durchdenken. In diesem Stadium erkannte Fehler und Unstimmigkeiten in den modellierten Prozessen sind noch recht leicht zu beheben. Ist ein Storyboard erst einmal erstellt, dann können die Entwickler leichter unabhängig voneinander auf ein gemeinsames Ziel hinarbeiten. Aber auch die Möglichkeit einen zukünftigen Anwender schon einmal mit dem Prototyp zu konfrontieren bringt für die Entwicklung ernorme Vorteile. Auch wenn bei der Entwicklung noch so viele Überlegungen angestellt wurden, so ist der spätere Anwender meist mit einigen Ausführungen nicht einverstanden und hätte diese gerne modifiziert. Auch hierfür gilt es so früh wie möglich dies zu erkennen, um damit einer späteren, meist aufwändigeren, Korrektur vorzubeugen.

8.1.1.2 Vorgehensweise

Zu Anfang fand die Idee ein Storyboard einzusetzen wenig Anklang. Das Vorgehen wurde zuerst als unnötige Arbeit eingeschätzt. Wir entschlossen uns schließlich das Storyboard in Form eines Microsoft PowerPoint Dokumentes zu verfassen, da die Benutzerführung auf diese Weise sehr leicht und flexibel modelliert werden konnte. Doch schon beim Erstellen traten Probleme auf, da viele Prozesse nicht detailliert genug modelliert waren. Bei der Umsetzung der Ideen und Visionen traten immer wieder Details an den Tag, die bisher einfach noch nicht bedacht wurden. Das Entfernen dieser vielen kleinen Unstimmigkeiten erforderte enormen zeitlichen Aufwand. Der Vorteil war jedoch, dass zu diesem Zeitpunkt noch keine Zeile Programmcode geschrieben war, so dass nur die Prozessmodellierung korrigiert werden musste.

8.1.1.3 Ergebnisse

Obwohl wir zurückblickend die Idee mit dem Storyboard nicht ganz konsequent und vor allem nicht bis zum Schluss umgesetzt haben, hat uns das Storyboard doch besonders im Vorfeld der eigentlichen Programmierung einen guten Dienst geleistet. Vor allem konnte auf diese Weise eine Menge späterer Anpassungsarbeit eingespart werden.

8.1.1.4 Probleme

Leider gibt es auch Nachteile, wenn man sich einmal dazu entschieden hat ein Storyboard zu verwenden. Trotz aller Mühe das Storyboard so flexibel und einfach wie möglich zu gestalten, ist der Aufwand beim Erstellen nicht zu unterschätzen. Auch der Umfang

entpuppte sich schnell als deutlich größer als vermutet. Schon bei dem Teil über den Übungsbetrieb kam schnell eine Ansammlung von mehr als 120 Seiten zustande. Die größten Schwierigkeiten hatten wir aber mit dem Versuch das Storyboard immer wieder zu aktualisieren und der tatsächlichen Entwicklung anzupassen. Der Aufwand wurde einfach zu groß und nach einigen eigentlich erfolglos zu nennenden Versuchen beschlossen wir, dass wir uns diese Arbeit nicht weiter machen würden, dazu diesem Zeitpunkt die eigentliche Arbeit der Programmierung bereits begonnen hatte.

8.1.2 JSP, Servlet und Struts-Framework

8.1.2.1 Beschreibung

JSP und Servlets

Servlets sind Module, die request/response-orientierte Server erweitern, beispielsweise könnte ein Servlet verantwortlich dafür sein, Daten eines HTML-Formulars zu entnehmen und diese dann an die Geschäftslogikschicht zur Verarbeitung weiterzureichen. Servlets werden vereinfacht gesagt als serverseitige Applets ohne graphische Oberfläche angesehen. Die JavaServer Pages-Technologie (JSP) [jsp] dient der dynamischen Erstellung webbasierter Inhalte mittels serverseitiger Abarbeitung. JSP vereinfacht den Prozess der Generierung dieser dynamischen Seiten durch Trennung der Applikationslogik vom Seitendesign und Kapselung der Logik in Portable, wiederverwendbare Java-Komponenten. Die JSP-Technologie hat sich aus der Servlet-Technologie heraus entwickelt und erweitert diese auf vielen Wegen, die Standardprozesse wie Building, Deployment und Maintaining vereinfachen; mit JSP gestaltete Seiten werden zu Servlets kompiliert.

Struts Framework

Das Open Source Produkt Struts [struts] des Jakarta Projekts [jak] der Apache Software Foundation [apache] bringt als flexibles Framework das Model-View-Controller-Prinzip MVC [mvc] in den Bereich der Webentwicklung. Es basiert im Wesentlichen auf den Techniken Java Servlets, JavaServer Pages, JavaBeans und Extensible Markup Language (XML) [xml].
Struts basiert auf dem MVC Model 2, einer Weiterentwicklung des ersten Entwurfes. In der MVC Model 1 Architektur war eine JSP-Seite allein verantwortlich für die Abarbeitung der eingehenden Anfrage vom Client und deren Beantwortung zurück an den Client. MVC Model 1 definiert zwar eine Trennung zwischen Content (View, implementiert durch JSP-Technologie) und Datenzugriff (Model, implementiert durch JavaBeans-Technologie), in komplexeren Projekten wird es jedoch zwangsläufig dazu kommen, dass Java Code notwendigerweise in dem JSP Code untergebracht werden muss. Dies ist für Java Programmierer zwar kein Problem, jedoch für JSP-Seiten, die von Designern erstellt und gewartet werden sollen, da diese in der Regel nicht über fundiertes Java-Wissen verfügen.
Die MVC Model 2 Architektur kombiniert die Servlet- und JSP-Technologie. Es werden die Vorteile und Stärken aus beiden Technologien gezogen und zusammengeführt: JSP-Seiten werden verwendet, um die Präsentationsschicht zu implementieren (View), Servlets, um prozessintensive Aufgaben durchzuführen (Controller). Das Servlet hat die Funktion eines Controllers, da es verantwortlich ist für die Request-Verarbeitung, für die Erstellung von JavaBeans oder Objekten, die in der JSP genutzt werden sollen, sowie für die Entscheidungsfindung, welche JSP-Seite als nächstes - in Abhängigkeit von Benutzer-eingaben - angezeigt werden soll. Ziel ist es, die Verarbeitungslogik komplett aus der JSP-Seite herauszunehmen und in den Controller zu verschieben.
Die Model-Komponente in Struts wird üblicherweise durch JavaBeans repräsentiert. Im Fall der PG nightshift jedoch ist das Model in den EntityBeans des Application Servers abgebildet, der Zugriff auf die EntityBeans erfolgt ausschließlich über SessionBeans. Für die View-Komponenten müssen sog. Forms und FormBeans erstellt werden, die Formulardaten sowie -validierungsmethoden (ausschließlich syntaktisch) enthalten. Controller-Komponenten in Struts müssen von der Klasse org.apache.struts.action.Action erben und sind verantwortlich für die vg. Aufgaben des Controllers in der MVC Model 2 Architektur.

8.1.2.2 Vorgehensweise

Im Rahmen der Projektgruppe nutzen wir Struts in der Version 1.1-b1. Die stabile Version 1.0 beinhaltete noch nicht alle Funktionen, die für die Umsetzung des Projekts notwendig waren. Beispielsweise war die Template-Engine „tiles“ noch nicht integriert und die Custom JSP Tag Libs waren noch nicht ausgereift.
Bei der Umsetzung der GUI war es unser Ziel, dass vorhandene Storyboard möglichst schnell und präzise in JSP-Seiten zu transformieren. Zu Anfang beschränkten wir uns auf reine Dummy-Seiten, d.h. auf die Erstellung der View-Komponenten. Für die Präsentationsschicht wurden mit der in Struts enthaltenen Template-Engine „tiles“ wieder verwendbare JSP-Seiten erstellt, um ein Standardlayout durch alle Seiten beizubehalten. Somit wurden ein einheitliches und konsistentes Erscheinungsbild gewährleistet. Nachdem die JSP-Seiten fertiggestellt waren, lag die Webseite in Rohfassung und ohne Funktionalität vor. Parallel wurde an den Businesslogik-Methoden (BL) gearbeitet, um sicherzustellen, dass an die BL-Methoden zu übergebende Parameter auch wirklich von der Oberfläche geliefert werden können. Nach Fertigstellung der JSP-Rohversion wurden auf die Formulare zugeschnittene ActionForm-Beans erstellt, Klassen mit Datentyp-Definition der Formulareingabefelder, die Formulardaten in JavaBeans-Form abbilden. Über Struts-spezifische Custom JSP Tags war es dann möglich, auf die Inhalte der ActionForm-Beans zuzugreifen. Die ActionForm-Beans werden von den sog. Action-Klassen mit Daten gefüllt. Die Implementierung der Action-Klassen war der nächste Schritt. Die Aufgaben der Action-Klassen bestehen darin, BL-Methoden aufzurufen und ermittelte Daten an den View weiterzugeben bzw. Daten aus Formularen an das Model weiterzureichen.

8.1.2.3 Ergebnisse

Hat man erst einmal die drei Komponenten Model, View und Controller im Sinne des Struts Frameworks implementiert, liegt einem ein stabiles, und nach subjektiver Beurteilung, sehr schnelles Arbeitsergebnis vor. Nach richtiger Planung eines Projektes ist man tatsächlich in der Lage, mehrere verschiedene View-Komponenten zu schaffen, die dieselben Controller und dasselbe Model nutzen. Zum größten Teil ist uns dies im Laufe der Projektarbeit auch gelungen, die View-Komponente „WAP“ nutzt etwa Action-Klassen der View-Komponente „HTML“ mit. Teilweise ließ sich jedoch eine Anpassung der Actionklassen im WAP-View aufgrund der WAP-Spezifikation (s. Kapitel WML Anwendung) nicht vermeiden.

8.1.2.4 Probleme

Probleme traten zu Anfang beim Verständnis des Struts-Konzeptes auf. Erste Experimente brachten nicht die gewünschten Ergebnisse. Beispielsweise wurde Controller-spezifischer Programmcode fälschlicherweise in FormBeans untergebracht etc. Diese logischen Fehler sind nicht unmittelbar aufgefallen, da der Code dennoch einwandfrei lief. Die teilweise recht eigenwillige Navigation durch die Dokumentation der Struts-Webseite sorgte oftmals für Verwirrung. Problematisch war auch der Aufwand für die Erstellung und Einbindung einer neuen View-Komponente:

- Erstellung der JSP-Seite
- Bekanntmachung der neuen Seite in der Konfigurationsdatei für Templates: tiles-defs.xml
- Erstellung einer FormBean
- Erstellung einer Action
- Bekanntmachung der FormBean in der Konfigurationsdatei von Struts: struts-config.xml.
- Verknüpfung der Action mit der FormBean sowie der JSP-Seite, ebenfalls in dieser Datei

Die beiden Konfigurationsdateien minderten durch ihren enormen Umfang die Übersichtlichkeit. Für Abhilfe sorgte das freie Tool Struts-Console [StrutsCon], dass die Konfiguration mit einer GUI ermöglicht. Ein vergleichbares Werkzeug für die tiles-defs.xml lag uns leider nicht vor. Zu Anfang betrug der Zeitaufwand für die Erstellung eines Prozesses, bestehend aus einem HTML-Formular und dessen Verarbeitung bei ca. 8 Stunden, dies konnte im Laufe des Projektes auf ca. die Hälfte reduziert werden.

8.1.3 WAP Anwendung

8.1.3.1 Beschreibung

Ein wichtiger Bestandteil der gestellten Aufgabe, war die Umsetzung der Geschäftsprozesse nicht nur für das Internet (Web-Client), sondern auch für eine mobile Plattform. Die Entscheidung fiel auf ein WAP-Portal (WAP-Client). Allerdings sollte die Funktionalität gegenüber dem Web-Client eingeschränkt werden. Es sollten nur solche Funktionen übernommen werden, die für die eingeschränkten Möglichkeiten eines Handys sinnvoll sind. Daher wurden für den WAP-Client nur die Gast- und die Studenten-Rolle umgesetzt. Dabei sollte genau wie beim Web-Client, die Struts-Technologie der Apache Group eingesetzt werden, wobei möglichst viel von der schon im Web-Client umgesetzten Software wiederverwendet werden sollte.

8.1.3.2 Vorgehensweise

Weil möglichst viel von der Web-Client-Umsetzung auch für den WAP-Client eingesetzt werden sollte, wurde mit der Umsetzung des WAP-Client erst im letzen Drittel der Projektzeit angefangen. Dabei wurden nach und nach die fertigen Web-Client-Prozesse auf das WAP-Portal überführt. Zunächst wurde die Gast-Rolle umgesetzt. Dann folgten Übungsgruppenmanagement und Prüfungsmanagement der Studentenrolle.

8.1.3.2.1 Technische Voraussetzungen für die Umsetzung des WAP-Client

Als Auslieferungsplattform diente auch hier ein Tomcat-Server. Damit konnte das Struts-Framework [struts] genauso eingesetzt werden wie im Web-Client. Zudem stand die komplette JSP-Technologie zur Verfügung. Um die umgesetzten WML-Seiten [wml] auch betrachten zu können, war ein WAP-Emulator notwendig. Eingesetzt wurde hier der „Deck-it previewer“ [Deck-it] von der Firma PyWeb.com. Dieser simulierte die Ausgaben auf einem Nokia 7110 Handy. Zudem standen uns leihweise einige Siemens ME45 Handys zur Verfügung. Diese wurden uns freundlicherweise von der Firma Siemens zu Testzwecken leihweise zur Verfügung gestellt. Die WML-Seiten sind nach dem WAP 1.0 Standard umgesetzt worden.

8.1.3.2.2 Umsetzung der WML-Anwendung

Zunächst wurden für das WAP-Portal eigene Unterverzeichnisse im Web-Client-Bereich geschaffen. Dadurch war es möglich, ohne weiteren Aufwand an Konfiguration oder Implementierung, die schon fertigen Struts-Module [struts] wieder zu verwenden. Aus der Sicht des Struts-Frameworks gab es also keinen Unterschied zwischen den beiden Clients.
Problematisch war allerdings die fehlende WML-Unterstützung [wml] von Struts. Daher musste als erstes eine eigene WML-Taglib geschrieben werden, um die Generierung von WML-Tags zu bewerkstelligen. Dazu wurde die vorhandene HTML-Taglib dupliziert und entsprechend umstrukturiert. Damit waren alle Voraussetzungen gegeben, um die eigentlichen WML-Seiten über das Struts-Framework generieren zu lassen. Die Struts-Konfiguration wurde nach und nach ergänzt und die WML-Seiten hinzugefügt. Dabei wurden möglichst viele Action- und Form-Klassen des Web-Clients mitgenutzt (z.B. bei der

Anmeldung und der Registrierung). Dies war allerdings, aufgrund der Einschränkungen eines Handys, nicht immer möglich.

8.1.3.3 Ergebnisse

Aufgrund der Wiederverwendung von Action- und Form-Klassen zur Kommunikation mit der Businesslogik war es möglich in recht kurzer Zeit den WAP-Client zu implementieren. Auch die Integration in die vorhandene Web-Client brachte einige Zeitersparnis.
Herausgekommen ist eine WAP-Client, der in der Bedienung, den entsprechenden Bereichen im Web-Client, sehr ähnlich ist. Es wurden nur die Bereiche umgesetzt, die für den Einsatz auf einem Handy auch sinnvoll erschienen. Da die WML-Seiten dem WAP 1.0 Standard entsprechen, sollte das Portal auf vielen Handys ausführbar sein. Allerdings leidet die Usability unter den Einschränkungen der Handyumgebung.

8.1.3.4 Probleme

Das erste Problem, was bei der Umsetzung auftauchte, war die fehlende WML-Taglib. Dies konnte aber relativ schnell durch eine eigene Implementierung behoben werden.
Ein weitaus größeres Problem war die Größenbeschrankung einer WML-Seite. Je nach Handy-Typ darf eine WML-Seite nur 1024 Byte (komprimiert) groß werden. Daher musste an einigen Stellen die Darstellung von Datenlisten auf mehrere WML-Seiten aufgeteilt werden. Dazu wurde eine eigene Action-Klasse erstellt, die automatisch die Datensätze aufteilt und einen „Blätter-Mechanismus“ beinhaltet.
Ein weiteres Problem war die geringe Auswahl an Formularelementen, die mit WAP 1.0 möglich sind. Es wurde nur Texteingabefelder und Selectboxen genutzt. Daher muss an einigen Stellen viel gescrollt werden.
Ein sehr großes Problem stellen leider immer noch die unterschiedlichen WAP-Browser-Implementierungen auf den verschiedenen Handys dar. Sogar Geräte von demselben Hersteller unterscheiden sich in der Darstellung der WML-Seiten. Daher wurden nur sehr wenige WML-Tags eingesetzt, von denen bekannt ist, dass sie auf möglichst vielen Handys funktionieren. Weiterhin wurde versucht, das Portal so zu gestalten, dass es bei der Darstellung der Inhalte auf möglichst vielen Handys zu keinen Problemen kommt. Zur Überprüfung dieses Aspekts wurden die Seiten auf mehreren Geräten getestet

8.2 Business Logic

8.2.1 EJB Komponenten Entwicklung

Das einzige in der PG-Ausschreibung festgelegte technische Kriterium für unsere Projektgruppe, war die komponentenorientierte Entwicklung der Anwendung mit Enterprise JavaBeans (EJBs). Diese von Sun Microsystems nun in der Version 2.0 verabschiedete Spezifikation [JEE2-v1.3] soll es einfach und vor allem standardisiert möglich machen, Software komponentenorientiert also „bausteinhaft“ zu entwickeln.
Im Folgenden soll nun zuerst diese Technologie ganz kurz umrissen werden. Danach werden wir darstellen, wie wir sie genutzt haben und in wie fern es sinnvoll war diese Technologie in diesem Umfang zu nutzen.

8.2.1.1 Beschreibung

Enterprise JavaBeans ist keine Software aus dem Hause Sun Microsystems, sondern lediglich eine Spezifikation also eine Vorgabe von Schnittstellen (Interfaces) und deren Verhalten. Implementiert werden diese von Herstellern von Application Servern, die dann wiederum von Sun für die Spezifikation einer bestimmten Version (zurzeit 2.0) zertifiziert werden. Danach entsprechen sie genau den Vorgaben von Sun und jeder, der sie als Applikationsplattform nutzt, soll sich darauf verlassen können. Somit soll es möglich sein portierbare Komponenten zu entwickeln, die theoretisch auf jedem Application Server laufen. Komponenten darf man sich dabei als eine Blackbox vorstellen, wovon man nur die nach außen sichtbaren Schnittstellen nutzt und das innere Verhalten kennt, aber nicht weiß, wie es implementiert ist. Mehrere dieser Komponenten, die nicht alle selbst entwickelt werden müssen, sondern auch zugekauft werden können, assembliert man zu einer vollständigen Applikation.
EJBs sind Teil der Java2 Enterprise Edition von Sun. Enterprise JavaBeans sind die Weiterentwicklung der bisherigen JavaBeans, erweitert mit Eigenschaften, die für die Entwicklung unternehmensbezogener Anwendungen fast immer von Bedeutung sind. Die wichtigsten sind:

- Persistenz
- Transaktionssicherheit
- Skalierbarkeit
- Modularität
- Integrierbarkeit
- Plattformunabhängigkeit
- Wiederverwendbarkeit

All diese Kriterien sollen durch den Einsatz von EJBs erleichtert und vor allem standardisiert werden. Es gibt drei Arten von EJBs die nun kurz mit ihrem Zweck erklärt werden.

- EntityBeans bilden die Persistenzschicht einer Applikation. Der Application Server ist bei der so genannten Container Managed Persistence (CMP) für das objektrelationale Mapping zwischen den Beans und einer Datenbank zuständig, womit der Entwickler von der Datenbankprogrammierung entlastet wird. Man kann EntityBeans auch mit Bean Managed Persistence (BMP) implementieren und die Abbildung der Daten in der Datenbank selbst programmieren.
- SessionBeans bilden die Applikationslogik einer Applikation. In ihr werden die Prozesse implementiert, die die Daten der EntityBeans verändern. Hier kommt besonders die Transaktionssicherheit zum Tragen, für die der Application Server automatisch sorgt.
- MessageDrivenBeans bilden die Schnittstelle zum Java Messaging Service (JMS) [jms] und somit zur Integration von Message Oriented Middleware (MOM). Eine MessageDrivenBean kann man nicht explizit aufrufen, sondern kann sie nur an eine Queue oder an ein Topic binden, und die Bean auf bestimmte Nachrichten reagieren lassen.

Mehrere EJBs kann man zu einer logischen Einheit zusammenbringen und als eine Komponente zur Verfügung stellen. Damit soll die gewünschte Modularität realisiert werden können. Aufgrund ihrer Unabhängigkeit vom System und weitgehend auch vom Application Server sind Komponenten wiederverwendbar und plattformunabhängig [JEE2-v1.3].

8.2.1.2 Vorgehensweise

Da die Entscheidung ob wir EJBs einsetzen sollten oder nicht, nicht anstand, richteten wir von Anfang an die Modellierung auf eine Entwicklung mit EJBs aus. Eine Spezifikation alleine reicht jedoch noch nicht aus, um eine Applikation zu modellieren.

8.2.1.2.1 Design Patterns

Um nicht alle typischen Schwierigkeiten und Herausforderungen beim Design selber bewerkstelligen zu müssen, wollten wir ein paar gängige Design Patterns einsetzen. Somit wurde einer Person aufgetragen, sich über Design Patterns zu informieren und einen Vortrag darüber zu halten. Danach entschieden wir uns für folgende Patterns und erklären kurz, warum wir das getan haben:

- „Session Facade“
  Session Facade bedeutet von Client-Seite keinen direkten Zugriff auf die EntityBeans (also die Persistenzschicht) zu haben, sondern ausschließlich über SessionBeans auf diese zuzugreifen. Besonders bei schreibenden Zugriffen stellte es sich als sehr nützlich heraus die Applikationslogik komplett auf dem Server zu behalten.
- „Value Objects“
  Value Objects sind Kopien der Inhalte einer EntityBean, die seriell über das Netz geschickt werden können. Hierdurch wird die Serverlast verringert, da nicht für jedes Attribut ein Aufruf an die Geschäftslogik gehen muss, sondern nur pro Objekt. An dieser Stelle haben wir aber noch eine andere Lösung zum Transport der Daten zum Client entwickelt, die im Abschnitt „DynaBeans“ weiter erläutert wird.

8.2.1.2.2 Modellierung

Die Modellierung begannen wir dann mit Hilfe von Interfaces, um nicht mit den schwergewichtigen Beans hantieren zu müssen. Wir planten dabei die Interfaces für die Beans, sowie bei den EntityBeans die zugehörige Value-Objekte, als gemeinsame Schnittstelle weiter zu verwenden. Dies gelang leider nicht aufgrund der EJB 2.0 Spezifikation, so dass wir nach dem Erstellen der Implementierungsklassen die Interfaces wieder löschten.

8.2.1.2.3 Entitätenschicht

Wir entwarfen zunächst die Entitätenschicht und teilten diese in Packages ein. Dabei machten wir reichlich Gebrauch von Container Managed Relationships zwischen den Beans, um die Relationen zwischen den Beans abbilden zu können. Leider beachteten wir dabei nicht, dass diese Vernetzung zur Folge hatte, dass wir die Beans nur alle zusammen zu einer Komponente hinzufügen konnten. Somit schied es von Anfang an aus, unsere Applikation in mehrere Komponenten aufzuteilen.

8.2.1.2.4 Geschäftslogik

Danach entwarfen wir wieder mit den Interfaces die Signaturen der Session Beans. Dabei wählten wir einen rollenbasierten Ansatz. Wir gaben jeder Rolle einen Controller, über den sie alle ihre zuständigen Geschäftsprozesse aufrufen können sollte. Dieses Konzept

behielten wir bis zu dem Zeitpunkt bei, als wir mehrere Methoden in unterschiedlichen Controllern doppelt implementieren mussten. Danach ordneten wir jedem Entitäten-Package jeweils einen Controller zu, wodurch sich Doppelimplementierungen vermeiden ließen.

8.2.1.3 Ergebnisse

Da EJB und damit die J2EE Technologie in Java-Kreisen derzeit das Thema Nr. 1 ist, war diese PG eine gute Gelegenheit sich mit dieser neuen Technologie auseinander zu setzen. Jedoch wollen wir dies einmal in den Hintergrund stellen und versuchen herauszufinden, was diese Technologie speziell für unser Projekt gebracht hat, denn die zwischenzeitliche Verzweiflung legte uns immer wieder Aussagen wie „Hätten wir doch bloß alles mit Perl gemacht“ oder ähnliches in den Mund.

Nachteile

- **Einarbeitungszeit und technische Probleme**
  Das wohl größte Problem war unsere Unerfahrenheit mit der neuen Technologie. Da einige von uns nicht einmal mit Java als Programmiersprache außerordentlich vertraut waren und einige die J2EE Technik auch nur aus der Theorie kannten, hatten wir natürlich entsprechende Probleme einen Prototyp unseres Systems erst einmal zu planen geschweige denn funktionierend umzusetzen. Dies benötigte allein etwa acht Wochen. Danach hatte etwa ein Drittel der Gruppe (also diejenigen, die dafür zuständig waren) ein grobes Verständnis dafür und konnte dieses praktisch umsetzten.
  Danach traten des Weiteren immer wieder Ungereimtheiten auf, die manchmal auch an unserem Application Server oder Together lagen und viel Zeit in Anspruch nahmen.
- **Einschränkungen beim Design**
  Wie schon erwähnt, planten wir z.B. Interfaces als gemeinsame Spezifizierung zu benutzen. Dies ließ der Application Server allerdings nicht zu. Ebenfalls fanden wir uns darin eingeschränkt, Objekte (Beans) nicht voneinander ableiten zu können.
- **Umstände beim Umgang**
  Ferner fanden wir es im Grunde umständlich drei Dateien (Bean, Home- und Remote-Interface) pflegen zu müssen.
- **Langer Deployvorgang**
  Als größter Nachteil bei der Entwicklung stellte sich allerdings der langwierige Deployvorgang, der in der Regel 1 Minute und länger dauerte, heraus. Dieser musste bei jeder kleinsten Änderung durchgeführt werden, was gerade in den Anfangsstadien der Entwicklung, wenn viel per Trial and Error herausgefunden und programmiert wird, zum Tragen kam.

Vorteile

Sieht man von den langen Einarbeitungszeiten ab, erscheint die ganze Technologie im Nachhinein natürlich nicht mal mehr halb so kompliziert, wie es sich anfangs darstellte. Im Wesentlichen kommen dann die Vorteile zutage mit denen die EJB Technologie an sich angepriesen wird. Besonders angenehm empfanden wir

- dass man sich nicht um Transaktionssicherheit kümmern musste
- die Abstraktion von der Datenbank, speziell die Container Managed Persistence und Relationship (CMP, CMR)
- die deklarative Zuweisung von Berechtigungen mit dem Java Authentification and Authorisation Service (JAAS)

Betrachtet man die Komplexität unseres Systems so kann man sicherlich erkennen, dass der Einsatz von EJBs die Nachteile, die sich durch die enorme Einarbeitung ergaben nicht mit den Vorteilen wettmachen konnte. Allerdings stehen bei einem Uni-Projekt auch nicht wirtschaftliche Faktoren im Vordergrund, so dass sich der Einsatz der EJBs zumindest

didaktisch gelohnt hat. Würde man nun im Anschluss ein ähnliches Projekt mit unserer bereits eingearbeiteten Gruppe starten sähe das wieder anders aus. Viele der Nachteile würden umgangen werden können. Zum anderen wächst die noch junge Technologie ständig weiter und verliert seine Kinderkrankheiten. Somit wäre ein erneuter Einsatz durchaus sinnvoll.

8.2.2 CMP

8.2.2.1 Beschreibung

Zwei Ausprägungen von EntityBeans sind in der Enterprise JavaBeans-Spezifikation von Sun vorgesehen: solche, bei denen die Persistenzlogik durch den Container implementiert wird (container managed persistence (CMP)) und solche bei denen diese in der Bean (selbst) zu implementieren ist (bean managed persistence (BMP)). Bei CMP wird die Persistenz einer EntityBean rein deklarativ festgelegt. Der Container entscheidet später aufgrund dieser Deklarationen, welche Funktionalität für die Persistenz generiert werden muss. Im Gegensatz dazu wird dies bei BMP selbst programmiert, d.h. in aller Regel wird die EntityBean über JDBC bzw. SQL/J persistent gemacht.

Während BMP bereits seit der Version 1.0 verbindlicher Teil der EJB-Spezifikation ist, ist dies für CMP erst ab der Version 1.1 der Fall. Insbesondere in den Bereichen Performanz und Portabilität hatte CMP in der Version 1.1 allerdings mit diversen Kinderkrankheiten zu kämpfen, die größtenteils auf eine unzureichende Spezifikation zurückzuführen waren. Erst mit dem Erscheinen der EJB 2.0- bzw. CMP 2.0-Spezifikation ist CMP als so weit gereift zu betrachten, dass sie als praxistauglich angesehen werden kann [GruSch02].

Eine Entscheidungsfindung für oder gegen eine der beiden Ausprägungen stützt sich auf deren Vor- bzw. Nachteile. Diese wiederum lassen sich anhand verschiedener Kriterien festmachen. Hierbei sei allerdings noch einmal darauf hingewiesen, dass eine solche Entscheidung pro EntityBean gefällt werden kann, also keineswegs für eine gesamte Anwendung gelten muss. Tabelle 1 zeigt eine mögliche Auswahl relevanter Kriterien und der damit verbundenen Fragestellungen.

Kriterium	Fragestellungen
Fehleranfälligkeit	Welche Fallstricke gibt es? Ist die Technik einfach zu verstehen?
Flexibilität	Wie umfangreich sind die „Ausdrucksmöglichkeiten“ mit der man die Persistenz beeinflussen kann? Kann ich ein bestehendes Datenbankschema weiternutzen?
Implementierungsaufwand	Wie zeitaufwändig ist die Umsetzung? Wie hoch ist der Einarbeitungsaufwand?
Wartbarkeit	Welche Konsequenzen haben Änderungen? Können Änderungen durch geeignete Mechanismen an den Nahtstellen abgefangen werden?
Performanz	Wie verhält sich eine Komponente bezüglich Durchsatz und Antwortzeit? Inwieweit ist die Performanz beeinflussbar?
Portabilität	Ist eine Komponente bzw. eine Anwendung von einem Application Server auf einen anderen portierbar? In welchem Umfang sind dazu gegebenenfalls Anpassungen notwendig?

Tabelle 1: Kriterien

Während die Antwort auf die Frage nach der Bewertung der Fehleranfälligkeit, der Flexibilität, des Implementierungsaufwands, der Wartbarkeit und der Portabilität leicht fällt,

da diese im wesentlichen im zugrunde liegenden Modell (und damit in der Spezifikation) selbst begründet sind, ist diese Frage für die Performanz nicht so einfach und allgemeingültig zu beantworten. Zum einen schreibt man der CMP hohe Performanz durch vorausschauende Optimierungen und Caching zu, zum anderen vertraut man eher den möglichen Handoptimierungen bei BMP. Zudem wird eine mögliche Antwort auf diese Frage im großen Maße vom gewählten Application Server beeinflusst und ist somit implementierungs- bzw. herstellerabhängig. Im Zweifelsfall bleibt hier wohl nur die Möglichkeit die beiden Varianten an der konkreten EntityBean gegeneinander zu messen. [GruSch02] schlägt dazu die Strategie vor, zunächst die EntityBeans möglichst mit CMP zu realisieren und diese später – abhängig von Bedarf und Messung – auf BMP umzustellen, „ohne dass große Aufwände verloren sind".
Die in der Tabelle 2 gezeigte bewertende Gegenüberstellung von CMP bzw. BMP der EJB 2.0-Spezifikation weist die Performanz für beide Varianten wegen der attestierten Praxistauglichkeit als „hoch" aus, klammert diese jedoch aus oben genannten Gründen ein.

	CMP	BMP
Fehleranfälligkeit	niedrig	hoch
Flexibilität	niedrig	hoch
Implementierungsaufwand	niedrig	hoch
Wartbarkeit	hoch	niedrig
Performanz	(hoch)	(hoch)
Portabilität	hoch	niedrig

Tabelle 2: CMP vs. BMP

Letztlich bewogen uns die hohe Portabilität, die niedrige Fehleranfälligkeit und der relativ niedrige Implementierungsaufwand dazu ausnahmslos CMP einzusetzen. Daher werden die Kernelemente dieser Ausprägung im Folgenden kurz beschrieben – für ausführlichere Erklärungen und entsprechend umfangreicheren Beispielen sei allerdings auf [RoAmJe02] verwiesen. Die Erläuterung gilt dabei für CMP in der Version 2.0, welche sich zum Teil beträchtlich von der Version 1.1 unterscheidet – eine detaillierte Aufstellung dieser Unterschiede findet man etwa in [Mon02].

Exemplarisch soll uns für die Erläuterungen ein einfaches Beispiel dienen, wie es auch in Clevery zu finden ist: ein Benutzer kann eine oder mehrere Benutzerrolle(n) innehaben und eine Benutzerrolle kann von einem oder mehreren Benutzer(n) ausgeübt werden. Auf die Clevery-Domäne bezogen heißt das etwa, dass eine Person gleichzeitig die Rollen *Student* und *Gruppenleiter* ausüben kann und dass die Rolle *Organisator* z.B. von den Personen *A* und *B* ausgeübt wird. Es handelt sich dabei also um eine m-zu-n Relation, wie man Abbildung 8-1 entnehmen kann.

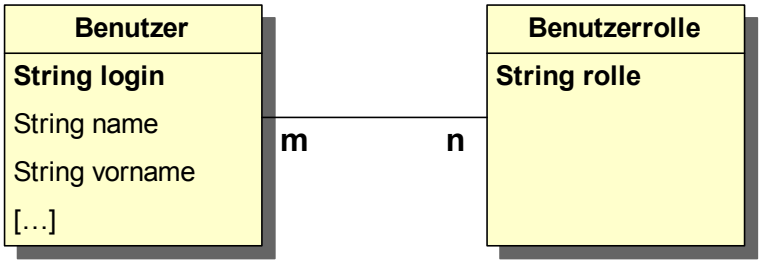


Abbildung 8-1: Überblick über das Beispiel „Benutzer / Benutzerrolle"

Was ist also zu tun, um die beiden Entitäten und ihre Relation mittels CMP abzubilden, d.h. welche Schritte sind dazu notwendig um zwei CMP-EntityBeans mit den entsprechenden Attributen und der entsprechenden Relation zu erhalten?

Zu jeder EnterpriseBean gehört zunächst einmal eine Bean-Klasse (ejb-class), eine Lebenszyklus- (home-interface) und eine Komponentenschnittstelle (component-interface) sowie eine Menge von Deklarationen im Deploymentdeskriptor. Eine EntityBean kann darüber hinaus eine Klasse für den Primärschlüssel verwenden, falls dieser beispielsweise aus mehr als einem Feld besteht.
Für unsere Betrachtung im Kontext CMP sind vor allem Bean-Klasse (Abbildung 8-2), Lebenszyklusschnittstelle (Abbildung 8-3) und Deploymentdeskriptor (Abbildung 8-4) interessant. Diese werden daher – beispielhaft für die EntityBean „Benutzer“ – näher betrachtet.

```
01: package de.pg_nightshift.server.user.entity;
02: […]
03: public abstract class BenutzerBean implements javax.ejb.EntityBean {
04:   private EntityContext entityContext;
05:   public void setEntityContext(EntityContext entityContext) {
06:     this.entityContext = entityContext;
07:   }
08:   public void unsetEntityContext() {
09:     this.entityContext = null;
10:   }
11:   public void ejbActivate() {}
12:   public void ejbPassivate() {}
13:   public void ejbRemove() {}
14:   public void ejbStore() {}
15:   public void ejbLoad() {}
16:   public String ejbCreate(String login) throws CreateException {
17:     setLogin(login);
18:     return null;
19:   }
20:   public void ejbPostCreate(String login) throws CreateException {}
21:   public abstract String getLogin();
22:   public abstract void setLogin(String login);
23:   public abstract String getName();
24:   public abstract void setName(String name);
23:   public abstract String getVorname();
24:   public abstract void setVorname(String vorname);
25:   […]
26:   public abstract Collection getRollen();
27:   public abstract void setRollen(Collection param);
28:   […]
29: }
```

Abbildung 8-2: Bean-Klasse der EntityBean „Benutzer“ (BenutzerBean.java)

Die Bean-Klasse ist das Herzstück einer EnterpriseBean, denn sie implementiert die eigentliche Funktionalität. Bei einer EntityBean, die CMP verwendet, muss diese Bean-Klasse zunächst einmal als abstrakte Klasse definiert werden, die die Schnittstelle `javax.ejb.EntityBean` implementiert (Abbildung 8-2, Zeile 3).
Im Deploymentdeskriptor müssen anfangs Name der Bean, sowie Schnittstellen und Bean-Klasse angegeben werden. Darüber hinaus sind EntityBean- bzw. CMP-spezifische Angaben notwendig: der Persistenz-Typ ist unserem Beispiel „Container“, die Klasse unseres Primärschlüssels ist „java.lang.String“, die CMP-Version ist „2.x“ und der abstrakte Schemaname, auf den wir später noch einmal zurückkommen werden, lautet „BenutzerBean“.
Um ein spezifisches Attribut einer EntityBean als CMP-Feld zu kennzeichnen, um es also als persistent zu markieren, muss für dieses Feld in der Bean-Klasse eine als `public abstract` gekennzeichnete getter bzw. setter Methode dazu angelegt werden. In unserem Beispiel wird dies für die Felder *Login, Name* und *Vorname* gezeigt (Abbildung 8-2, Zeilen 21-24). Im Deploymentdeskriptor müssen diese Felder ebenfalls angegeben werden (Abbildung 8-4, Zeilen 15-23). Das Feld eines einfachen Primärschlüssels, in unserem Beispiel das Feld *Login*, muss dabei zusätzlich gesondert gekennzeichnet werden (Abbildung 8-4, Zeile 25).
Die Relation zwischen Benutzer und Benutzerrolle wird ebenfalls über den EJB Container verwaltet. Man spricht dabei von „container managed relationships“ oder kurz CMR. Analog zu unseren CMP-Feldern enthält unsere Bean-Klasse dazu ein CMR-Feld „Rollen“, welches die entsprechende Relation darstellt. Der Anteil im Deploymentdeskriptor der die Relation beschreibt ist hier wesentlich umfangreicher. Neben dem Namen der Relation werden hier für beide Relationsrollen der Rollenname, die Kardinalität, die Relationsquelle und das Relationsfeld angegeben. Wie man der Abbildung entnehmen kann, ist für die

„Benutzerrollen-Seite“ kein solches Feld angegeben. Man hat also für jeden Benutzer seine zugeordneten Rollen über das gleichnamige Feld im Zugriff, der umgekehrte Zugriff von einer Rolle auf die zugewiesenen Benutzer ist jedoch nicht möglich. Eine solche Relation nennt man daher auch unidirektional.
Wie bekommt man beispielsweise alle Benutzer zu einer bestimmten Rolle? Die Antwort dazu lautet: „finder“-Methoden mittels EJB QL zu deklarieren.
EJB QL ist die Abfragesprache für EntityBeans – ihr Aufbau erinnert stark an SQL bzw. OQL, ihr Umfang dagegen weniger, da sie zwar Selektionsabfragen, aber keine Aktualisierungs- oder Löschoperationen ermöglicht. EJB QL ist von der konkreten Persistenzimplementierung unabhängig und arbeitet auf einem so genannten abstrakten Schema. Dieses abstrakte Schema setzt sich aus dem abstrakten Schemanamen (Abbildung 8-4, Zeile 14) und den CMP/CMR-Feldnamen einer EntityBean zusammen, die EJB QL ist daher auch nur im Zusammenhang mit CMP/CMR nutzbar.
Um eine „finder“-Methode zu deklarieren gibt man zunächst ihre Signatur in der Lebenszyklusschnittstelle an (Abbildung 8-3, Zeilen 5-6). Dann formuliert man die zugehörige EJB QL-Abfrage und trägt diese im Deploymentdeskriptor ein (Abbildung 8-4, Zeilen 26-36).

```
01: package de.pg_nightshift.server.user.entity;
02: […]
03: public interface BenutzerLocalHome extends javax.ejb.EJBLocalHome {
04:   […]
05:   public Collection findByRolle(java.lang.String rolle)
06:     throws javax.ejb.FinderException;
07:   […]
08: }
```

Abbildung 8-3: Lebenszyklusschnittstelle der EntityBean „Benutzer“

```
01: […]
02: <ejb-jar>
03:  <enterprise-beans>
04:   […]
05:   <entity>
06:    <ejb-name>Benutzer</ejb-name>
07:    <local-home>de.pg_nightshift.server.user.entity.BenutzerLocalHome</local-home>
08:    <local>de.pg_nightshift.server.user.entity.BenutzerLocal</local>
09:    <ejb-class>de.pg_nightshift.server.user.entity.BenutzerBean</ejb-class>
10:    <persistence-type>Container</persistence-type>
11:    <prim-key-class>java.lang.String</prim-key-class>
12:    <reentrant>False</reentrant>
13:    <cmp-version>2.x</cmp-version>
14:    <abstract-schema-name>BenutzerBean</abstract-schema-name>
15:    <cmp-field>
16:     <field-name>login</field-name>
17:    </cmp-field>
18:    <cmp-field>
19:     <field-name>name</field-name>
20:    </cmp-field>
21:    <cmp-field>
22:     <field-name>vorname</field-name>
23:    </cmp-field>
24:    […]
25:    <primkey-field>login</primkey-field>
26:    <query>
27:     <query-method>
28:      <method-name>findByRolle</method-name>
29:      <method-params>
30:       <method-param>java.lang.String</method-param>
31:      </method-params>
32:     </query-method>
33:     <ejb-ql>
34:      SELECT OBJECT(b) FROM BenutzerBean b, IN (b.rollen) r WHERE r.rolle = ?1
35:     </ejb-ql>
36:    </query>
37:    […]
38:   </entity>
39:   […]
40:  </enterprise-beans>
41:  <relationships>
42:   <ejb-relation>
43:    <ejb-relation-name>BenutzerRollen</ejb-relation-name>
44:    <ejb-relationship-role>
45:     <ejb-relationship-role-name>BenutzerHabenRollen</ejb-relationship-role-name>
46:     <multiplicity>Many</multiplicity>
47:     <relationship-role-source>
48:      <ejb-name>Benutzer</ejb-name>
49:     </relationship-role-source>
50:     <cmr-field>
51:      <cmr-field-name>rollen</cmr-field-name>
52:      <cmr-field-type>java.util.Collection</cmr-field-type>
53:     </cmr-field>
54:    </ejb-relationship-role>
55:    <ejb-relationship-role>
56:     <ejb-relationship-role-name>RollenZuBenutzern</ejb-relationship-role-name>
57:     <multiplicity>Many</multiplicity>
58:     <relationship-role-source>
59:      <ejb-name>Benutzerrolle</ejb-name>
60:     </relationship-role-source>
61:    </ejb-relationship-role>
62:   </ejb-relation>
63:   […]
64:  </relationships>
65:  […]
66: </ejb-jar>
```

Abbildung 8-4: Deploymentdeskriptor (ejb-jar.xml)

8.2.2.2 Vorgehensweise

Zunächst setzten wir im Entwurf die in der OOA (Objektorientierte Analyse) identifizierten Typen in Java-Schnittstellen, und entsprechenden Beziehungen unter ihnen, um. Diese Schnittstellen dienten uns dann fast ausnahmslos 1-zu-1 als Grundlage für unsere CMP-EntityBeans. Felder wurden so zu CMP-Feldern und Assoziationen, Aggregationen sowie Kompositionen zu Relationen bzw. CMR-Feldern.
Für die Entitätenschicht stellte sich dieses Vorgehen als ausreichend heraus, da der gewählte Entwurf bis zur Fertigstellung weitgehend stabil blieb. Zu diesem Umstand mag auch beigetragen haben, dass wir uns frühzeitig darauf einigten, die EntityBeans frei von jeder Geschäftlogik zu halten und sie entsprechend als reine Datenobjekte zu sehen.

Als Werkzeug zur Modellierung, Programmierung und zum Deployvorgang setzten wir zunächst Togethersoft ControlCenter ein, welches unter anderem die Möglichkeit bietet den Deploymentdeskriptor aus Modellen bzw. Diagrammen zu generieren. Nachdem sich die zunächst verwendete Version 5.5 als nicht EJB 2.0 bzw. CMP 2.0 konform erwies – und damit für uns unbrauchbar wurde – arbeiteten wir mit der Version 6.0 weiter. Dies bedeutete zwar zuerst einen Fortschritt, aber auch die Version 6.0 lässt bis heute bei der Unterstützung der EJB-Spezifikation zu wünschen übrig. Dieser Umstand brachte uns schließlich zu xdoclet, einem auf JavaDoc basierenden Werkzeug zur Codegenerierung (u. a. Deploymentdeskriptoren und Schnittstellen), mit dem es uns schlussendlich gelang, das Potential von EJB bzw. CMP auszuschöpfen und den Deployvorgang sowohl zu vereinfachen als auch gleichzeitig zu beschleunigen.
Da wir, wie bereits erwähnt, reichlich Gebrauch von CMP bzw. CMR machten, waren wir entsprechend auf einen diesbezüglich konformen Application Server angewiesen. JBoss, unser frühzeitig auserkorener Favorit unter den Application Servern, unterstützte in der damals aktuellen Version 2.4 kein CMP 2.0. So suchten wir nach Alternativen und wurden schließlich im MVCSoft PersistenceManager fündig. Der MVCSoft PersistenceManager bietet die entsprechende Persistenzfunktionalität und lässt sich im Zusammenspiel mit diversen Application Servern, darunter JBoss 2.4, betreiben. Außerdem bietet der PersistenceManager eine grafische Oberfläche zum objektrelationalen Mapping und umfangreiche Funktionalität zum „Fein-Tuning“ desselbigen. Leider blieb uns zum Ende des Projekts nicht mehr die Zeit, die Möglichkeiten des Werkzeugs zu ergründen und zu nutzen.

8.2.2.3 Probleme

Auch wenn wir die Technik zum Ende des Projekts zunehmend in den Griff bekamen, blieben einige Probleme, die insbesondere in der EJB bzw. CMP Spezifikation selbst begründet sind. Dazu zählen:

8.2.2.3.1 keine Vererbung/Polymorphie

Die EJB Spezifikation sieht keine Vererbung und keine Polymorphie vor – zumindest nicht im herkömmlichen Sinne, d.h. etwa über das `extends`-Schlüsselwort in einer JAVA-Klasse. Da EntityBeans insbesondere auch eine objektorientierte Sicht auf die Daten der Domäne liefern soll, ist dieser Umstand sehr unbefriedigend. Daniel O´Connor zeigt in [OCo02] wie man dennoch Vererbungs- bzw. Polymorphieverhalten über Delegation nachbilden kann.

8.2.2.3.2 Fehlendes „ORDER BY” in EJB QL

Ergebnismengen einer finder-Methode können nicht sortiert zurückgegeben werden, da eine „ORDER BY“-Klausel wie in SQL fehlt. Man kann diese Problematik umgehen in dem man entweder Client-seitig in der Darstellung, oder Server-seitig, z.B. in einer vor der EntityBean geschalteten SessionBean, programmatisch sortiert. Die EJB 2.1-Spezifikation [Sun02] soll hier Abhilfe schaffen.

8.2.2.3.3 Fehlendes „UPPERCASE“ in EJB QL

Vergleiche in der „WHERE-Klausel“ einer EJB QL-Abfrage berücksichtigen, aus Ermangelung einer „UPPERCASE“ bzw. „LOWERCASE“ Funktion, immer Groß-/Kleinschreibung. Ist man auf entsprechende Vergleiche angewiesen, z.B. bei Suchfunktion mit Benutzereingabe, so ist dies, möchte man portabel bleiben, nur über BMP möglich.

8.2.2.3.4 Keine automatisch generierten Primärschlüssel

Die EJB 2.0-Spezifikation sieht keine Möglichkeit vor, CMP-Primärschlüssel als „autogeneriert“ zu deklarieren. Eine solche Funktionalität wird häufig benötigt und eine Erweiterung der Spezifikation dahingehend wäre entsprechend sinnvoll.

8.2.2.3.5 Komplexer Deployvorgang

Auch kleine Änderungen an den Feldern oder Relationen einer EntityBean bedeuteten, dass sich das zugrunde liegende Datenbankschema mit änderte und der Deployvorgang unserer Anwendung entsprechend komplex und langwierig wurde. Hier wäre entsprechende Unterstützung durch ein Werkzeug wünschenswert.

8.2.2.4 Ergebnisse

Wie bereits eingangs erwähnt, wurden alle EntityBeans in unserem Projekt mit CMP umgesetzt. Da wir weder auf bestehenden Daten aufsetzen mussten bzw. konnten, noch in einer anderen Weise hohe Flexibilität bei der Umsetzung der Persistenzlogik vonnöten gewesen ist, wäre diese Entscheidung wohl auch aus heutiger Sicht noch genauso zu treffen.
Der vermeintliche geringere Implementierungsaufwand von CMP gegenüber BMP, auf den wir zu Beginn setzten, kann sich jedoch als Trugschluss erweisen, speziell dann, falls bereits Know-how im Bereich JDBC vorhanden ist und im CMP Umfeld (noch) nicht. Der deklarative Ansatz von CMP ist nämlich keinesfalls so intuitiv als dass er ohne Einarbeitungszeit umzusetzen wäre.
Hat man die Technologie allerdings erst einmal weitgehend im Griff, so ist die Verwendung von CMP, aus unserem derzeitigen Kenntnisstand heraus, sinnvoll und empfehlenswert zu nennen. Im Bereich der Flexibilität von CMP sind dabei sicherlich noch einige Wünsche offen (siehe Probleme), insgesamt ist die Funktionalität für ein Informationssystem, wie es Clevery darstellt, aber ausreichend.

8.2.3 Deployment Tool

8.2.3.1 Beschreibung

Das „JBoss Deployer Plugin for Together ControlCenter“ von M. J. Swainston-Rainford [DepTool] Version 1.0.6 ermöglicht das Deployen von J2EE Applikationen auf dem JBoss Application Server direkt aus Together ControlCenter heraus. Es generiert hierfür die JBoss spezifischen XML Deskriptoren in den Dateien jboss.xml, jboss-web.xml und jaws.xml. Um die benötigten Informationen abzufragen integriert es neue Properties-Dialoge in Together, hierbei wird auch der MVCSoft Persistence Manager integriert [mvc].
Darüber hinaus ermöglicht das Tool das Starten und das Stoppen von JBoss und behebt einige die EJB 2.0 Spezifikation betreffende Bugs von Together.

8.2.3.2 Vorgehensweise

Die Installation des Plugins gestaltet sich sehr einfach. Es muss lediglich eine ZIP-Datei in das Together Verzeichnis entpackt werden. Der JBoss 2.4 steht dann als Option im „Deployment Expert“ von Together zur Verfügung. Mit den erweiterten Properties-Dialogen können die JBoss spezifischen Einstellungen bearbeitet werden. Hier kommen vor allem für die CMP 2.0 in Verbindung mit dem MVCSoft Persistence Manager viele Optionen hinzu.
Beim eigentlichen Deployvorgang werden die benötigten XML-Dateien erzeugt und können gegebenenfalls nachbearbeitet werden. Außerdem wird hier automatisch die Codegenerierung durch den MVCSoft Persistence Manager durchgeführt. Am Ende wird die generierte JAR- bzw. EAR-Datei automatisch in das JBoss Deploy Verzeichnis kopiert.
Obwohl das Plugin eigentlich für Together 5.5 geschrieben wurde, konnten wir es auch mit Together 6.0 einsetzten. Hierzu musste lediglich eine JAR-Datei (ejb20fixup.jar), die einige Bugs in Together 5.5 korrigiert, aus dem Installationsverzeichnis gelöscht werden.

8.2.3.3 Ergebnisse

Insgesamt waren wir mit dem Plugin sehr zufrieden. Leider behebt das Tool aber nicht das Problem, dass manuelle Nachbearbeitungen des Deploymentdeskriptors nicht gesichert werden können.
Durch die Unzulänglichkeiten von Together, insbesondere bei Verwendung von JAAS, wurde das Plugin für uns gegen Ende des Projektes überflüssig, da wir eine komplett von Together unabhängige Lösung zum Deployen wählen mussten, die Deploymentdeskriptor sowie Local-, Home- und Remoteinterfaces automatisch aus Kommentaren im Beancode generiert. Es handelte sich dabei um die OpenSource Software XDoclet, die bei Sourceforge herunter geladen werden kann.

8.2.4 Sicherheitskonzept JAAS

8.2.4.1 Beschreibung

JAAS ist die Abkürzung für „Java Authentification and Authorization Service“ und bezeichnet die Standardimplementierung des deklarativen Sicherheitsmodells, das durch die Verwendung von Login-Modulen und die Verwaltung von Subjects umgesetzt wird.
JAAS ist in das JBoss Security Extension Framework, kurz JBossSX, eingebunden, das neben JAAS noch Security Proxy Layers unterstützt. Die Security Proxy Layer haben aber den Nachteil, dass die Sicherheitseinstellungen nicht über ein deklaratives Modell für eine EJB festgelegt werden können, sondern im EJB Business Object modelliert werden müssen. Aus diesem Grund wird für Clevery das deklarative Modell verwendet.

8.2.4.1.1 Übersicht über die deklarative Sicherheit in J2EE

Das in der J2EE spezifizierte Sicherheitsmodell ist ein deklaratives Modell, das die Aspekte und die Umsetzung von Sicherheitsmechanismen weitgehend vom Code trennt. Die Idee Sicherheitscode aus dem Businesslevelcode herauszuhalten, liegt darin begründet, dass die Realisierung von Security-Mechanismen eher eine Aufgabe der lokalen Deploy-Umgebung ist, als ein inhärenter Aspekt der Businesslogik. Die Sicherung einer J2EE Application basiert auf der Spezifikation der Anwendungssicherheitsanforderungen per Standard J2EE-Deploymentdeskriptoren.

8.2.4.1.2 Die wichtigsten Elemente bei der Sicherung

Die wichtigsten Bestandteile der JAAS Implementierung sind:
- Security References
- Security Identity
- Security Roles
- EJB Method Permissions
- Web Content Security

Mit diesen Konstrukten ist es möglich ein Sicherheitskonzept mit JAAS umzusetzen. Aus diesem Grund erfolgt an dieser Stelle eine Beschreibung dieser Komponenten bevor genauer auf das JAAS-Konzept, welches in Clevery umgesetzt wurde, eingegangen wird.

8.2.4.1.3 Security Referenzen

Sowohl EJBs als auch Servlets können ein oder mehrere „security-role-ref“-Elemente deklarieren. Diese werden benötigt um zu deklarieren, dass eine Komponente den „role-name“-Wert als ein Argument für die „isCallerInRole(String)“-Methode nutzt. Mit dieser Methode ist eine Komponente dann in der Lage zu überprüfen, ob ein „Caller“ eine Rolle innehat, die mit Hilfe eines „security-rol-ref/role-name“-Element deklariert wurde. Das „role-name“-Element muss per „role-link“ auf ein „security-role“-Element verlinkt werden. Die Hauptanwendung der „isCallerInRole“-Methode ist die Ausführung einer Sicherheits-

überprüfung, die nicht durch die Vergabe von Methodenrechten („method permissions“) modelliert werden kann. Ein einfaches Beispiel [JBDoc] für die Deploymentdeskriptoren:

```
<!-- A sample ejb-jar.xml fragment -->
<ejb-jar>
<enterprise-beans>
        <session>
             <ejb-name>ASessionBean</ejb-name>
   ^^              ...
<security-role-ref>
                    <role-name>TheRoleICheck</role-name>
                    <role-link>TheApplicationRole</role-link>
           </security-role-ref>
      </session>
</enterprise-beans>
...
</ejb-jar>

<!-- A sample web.xml fragment -->
<web-app>
<servlet>
<servlet-name>AServlet</servlet-name>
...
<security-role-ref>
                  <role-name>TheServletRole</role-name>
                  <role-link>TheApplicationRole</role-link>
</security-role-ref>
</servlet>
...
</web-app>
```

Abbildung 8-5: Deploymentdeskriptor

8.2.4.1.4 Security Identity

Für EJBs kann optional ein „security-identity element“ deklariert werden. Neu bei EJB 2.0 ist die Möglichkeit zu spezifizieren welche Identität eine EJB annehmen soll, wenn sie auf die Methoden anderer Komponenten zugreift. Die „invocation identity“ kann entweder die des aktuellen „Caller“ sein, oder auch eine spezifische Rolle. Der Application Assembler nutzt das „security-identity“ Element zusammen mit dem „use-caller“ Childelement um anzuzeigen, dass die Identität des aktuellen „Callers“ als Security Identität für Methodenaufrufe der EJB benutzt werden soll. Der Übertrag der „Caller“ Identität ist das Standardvorgehen, falls es keine explizite Deklaration einer „security-identity“ gibt.

```
<!-- A sample ejb-jar.xml fragment -->
ejb-jar>
<enterprise-beans>
        <session>
                <ejb-name>ASessionBean</ejb-name>
                  ...
<security-identity>
<use-caller-identity/>
                </security-identity>
             </session>
           <session>
                <ejb-name>RunAsBean</ejb-name>
                   ...
                <security-identity>
                 <run-as>
                       <description>A private internal role</description>
                       <role-name>InternalRole</role-name>
                </run-as>
                </security-identity>
       </session>
</enterprise-beans>
...
</ejb-jar>
```

Abbildung 8-6: „ejb-jar.xml“ Deskriptor

Alternativ kann der Application Assembler auch das „run-as/role-name“- Childelement nutzen um zu spezifizieren, dass eine spezifische Security Role, die durch den „role-name“ Wert festgelegt ist, als Security-Identität genutzt werden soll. Wichtig dabei ist, dass dadurch nicht die Identität des „Callers“ verändert wird, die durch „EJBContext.getCallerPrincipal()“

einsehbar ist. Genau genommen werden die Security-Rollen des „Callers“ auf die eine Rolle gesetzt, die durch den Wert des „run-as/role-name“ Elements angegeben wurde. Ein Anwendungsgebiet für das „run-as“ Element ist es zu verhindern, dass externe Clients auf interne EJB zugreifen können. Dies wird dadurch erreicht, indem den internen EJBs „method permissions“ zugewiesen werden, die den Zugriff nur auf Rollen begrenzen, die nicht einem externen Client zugewiesen sind. EJBs, die eine interne EJB nutzen müssen, werden dann mit Hilfe von „run-as/role-name“ entsprechend konfiguriert. Ein Beispiel für einen „ejb-jar.xml“ Deskriptor ist in Abbildung 8-6 dargestellt [JBDoc].

8.2.4.1.5 Security Roles

Der Security Role Name, referenziert durch das Element „security-rol-ref“ oder „security-identity“, muss auf eine in der Applikation deklarierten Rolle abgebildet werden. Der Application Assembler definiert die logischen Security Roles durch die Deklarierung des „security-role“ elements. Der Wert „role-name“ ist ein logischer Rollenname in der Applikation wie z.B. Administrator, Gast etc.
In der J2EE-Spezifikation ist festgehalten, dass es wichtig ist darauf zu achten, dass die Security-Rollen im Deploymentdeskriptor dazu benutzt werden, um die logische Security-Sicht auf eine Applikation zu definieren. Sie sollten nicht verwechselt werden mit Usergroups, User, Principals und anderen Konzepten, die im „Target Enterprise's operational Environment“ existieren.
Beim JBoss wird eine Security Rolle lediglich dazu genutzt die Werte von „security-role-ref/role-name“ auf die logische Rolle zu mappen, zu welcher der in der Komponente deklarierter Rollenname gehört. Die dem Benutzer zugewiesenen Rollen sind eine dynamische Funktion des Security Managers der Applikation. JBoss verlangt bei der Definition von Security-Rollen keine bestimmte Reihenfolge, um die Rechte für die Methoden zu deklarieren. Ein Beispiel[JBDoc] für die Nutzung von „security-role“:

```
<!-- A sample ejb-jar.xml fragment -->
<ejb-jar>
...
<assembly-descriptor>
     <security-role>
         <description>The single application role</description>
         <role-name>TheApplicationRole</role-name>
     </security-role>
</assembly-descriptor>
</ejb-jar>

<!-- A sample web.xml fragment -->
<web-app>
...
  <security-role>
       <description>The single application role</description>
       <role-name>TheApplicationRole</role-name>
  </security-role>
</web-app>
```

Abbildung 8-7: Security Roles im „ejb-jar.xml“ Deskriptor

8.2.4.1.6 EJB Method Permissions

Im Application Assembler können die Rollen festgelegt werden, denen erlaubt ist die Methoden des EJB Home- und Remote-Interface aufzurufen. Dies wird umgesetzt indem ein „method-permission“ Interface deklariert wird. Jedes „method-permission“ Element beinhaltet ein oder mehrere „role-name“ Childelemente, die definieren wie die logischen Rollen Zugriff auf EJB Methoden haben. Festgelegt wird dies durch die „method“ Childelemente. Ab der Version EJB 2.0 ist es möglich ein „unchecked“ Element anstelle eines „role-name“ Elements anzugeben um darzustellen, dass jeder authentifizierte User auf die Methoden zugreifen kann die durch das „method“ Childelement ausgezeichnet wird. Zusätzlich ist es möglich auch über das „exclude-list“ Element jeglichen Zugriff auf Methoden zu unterbinden.

Ein sehr wichtiger Aspekt ist, dass eine EJB Methode defaultmäßig nicht benutzt werden kann, wenn sie zuvor nicht durch ein „method permission“ Element für eine Rolle freigegeben wurde. Beispiele für die Einträge in der „ejb-jar.xml“ Datei:

```
<method-permission>
         <description>The employee role may access the
            findByPrimaryKey, getEmployeeInfo, and the
            updateEmployeeInfo(String) method of the
               AardvarkPayroll bean
         </description>
        <role-name>employee</role-name>
        <method>
           <ejb-name>AardvarkPayroll</ejb-name>
           <method-name>findByPrimaryKey</method-name>
         </method>

         <method>
            <ejb-name>AardvarkPayroll</ejb-name>
            <method-name>getEmployeeInfo</method-name>
         </method>

<method>
      <ejb-name>AardvarkPayroll</ejb-name>
      <method-name>updateEmployeeInfo</method-name>
      <method-params>
            <method-param>java.lang.String</method-param>
      </method-params>
   </method>
</method-permission>
```

Abbildung 8-8: Method Permissions im „ejb-jar.xml“ Deskriptor

8.2.4.1.7 Web Content Security Constraints

In einer Webapplikation ist Sicherheit durch die Rollen definiert, die auf einen Content, der durch ein URL-Pattern eingegrenzt ist, zugreifen dürfen. Die Festlegung dieser Parameter ist Gegenstand der Datei „web.xml“ und der in ihr befindlichen „security-constraint“-Elemente. Der zu sichernde Content ist durch das „web-resource-collection“-Element deklariert. Jedes „web-resource-collection“ Element beinhaltet eine optionale Serie von „url-pattern“-Elementen gefolgt von einer optionalen Serie von „http-method“-Elementen. Das „url-pattern“-Element spezifiziert ein URL-Pattern für einen gesicherten Bereich. Eine angeforderte URL wird mit diesem Pattern verglichen, so dass der Application Server den Zugriff auf einen restringierten Bereich erkennen kann (angeforderte URL entspricht dem Pattern). Das „http-method“ Element spezifiziert die erlaubten HTTP-Anfragen (get, put...).

Das optionale „user-data-constraint“ Element spezifiziert die Anforderungen an den Transport Layer der Client-Server Verbindung.

Das optionale „login-config“ wird benutzt, um die Authentifizierungsmethoden zu konfigurieren, die der Server benutzen soll. Dieselben Aufgaben erfüllen der Realm Name und die Attribute, die für den Form Login-Mechanismus benötigt werden. Das „auth-method“-Childelement spezifiziert die Authentifizierungsmechanismen für die Webapplikation. Als Voraussetzung für den Zugriff auf eine beliebige gesicherte Webresource muss der Benutzer durch die eingerichteten Mechanismen authentifiziert werden. Eine Beispiel Konfiguration zeigt Abbildung 8-9:

```
<web-app>
...
<security-constraint>
<web-resource-collection>
        <web-resource-name>Secure Content</web-resource-name>
        <url-pattern>/restricted/*</ url-pattern></
    <web-resource-collection>
    <auth-constraint>
        <role-name>AuthorizedUser</role-name>
    </auth-constraint>
    <user-data-constraint>
        <transport-guarantee>NONE</transport-guarantee>
    </user-data-constraint>
</security-constraint>
...
<login-config>
    <auth-method>BASIC</auth-method>
    <relam-name>The Restricted Zone</realm-name>
</login-config>
...
<security-role>
    <description>The role required to access restricted content
    </description>
    <role-name>AuthorizedUser</role-name>
</security-role>
</web-app>
```

Abbildung 8-9: Web Content Security Constraints in der Datei „web.xml“

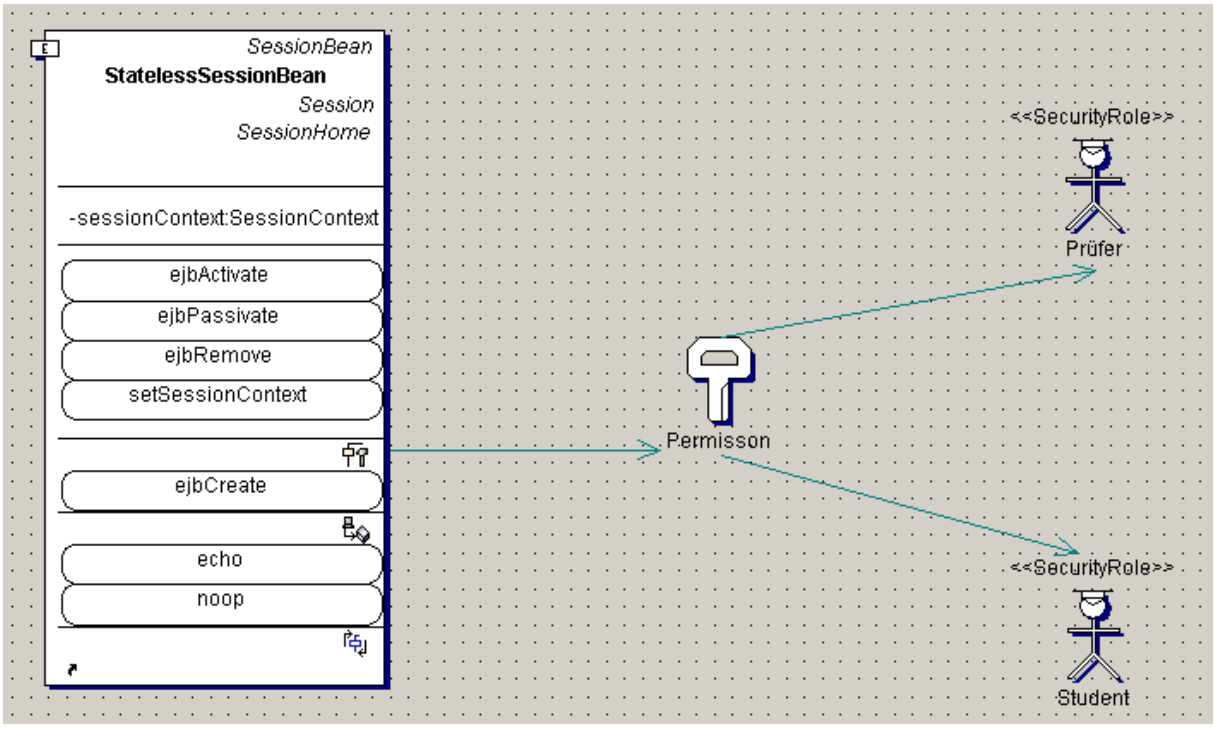


Abbildung 8-10: Einfaches EJB Assembler Diagramm

8.2.4.2 Vorgehensweise

Zentral für die Sicherung von Zugriffen auf EJBs und Web Komponenten in Enterprise Applikationen sind die Dateien „ejb-jar.xml“ und „web.xml“. In ihnen werden die Zugriffsrechte festgelegt.
Die Zuweisung von Rollen und entsprechenden Rechten sollte ursprünglich direkt mit Together verwirklicht werden. Das Verfahren dafür ist an sich recht einfach und intuitiv. In den beiden Diagrammen EJB Assembler Diagram und Web Application Diagram können direkt Method Permissions (Schlüssel Symbole) mit den Beans assoziiert werden. Diese Assoziation kann sich dabei auf einzelne Komponenten der Klasse beziehen oder auch auf

die Klasse als Ganzes und wird durch Assembly Links umgesetzt (von der Klasse/Methode zur Method Permission). Entsprechend der Rollen, die in der Applikation auftauchen (hier Student, Prüfer etc.), werden zusätzlich Security Roles in das Diagramm eingefügt und über Assembly Links mit den Method Permissions verbunden. Der Assembly Link verläuft hier von der Method Permission zur Security Role. Im Gegensatz zur Method Permission muss die Security Role einen eindeutigen Namen tragen. Über diesen Namen kann dann eine Autorisierung abgewickelt werden (siehe Abbildung 8-10).

Als letzter Schritt muss noch in den Properties für das Assembler Diagram der JNDI Name der Security Domäne angegeben werden. Über diesen Parameter ist dann der JBoss Server in der Lage die Art und Weise der Authentifizierung zu erkennen und anzuwenden. Mit diesen Daten und Parametern erzeugt Together während des Deploy-Vorgangs die Deskriptordateien „ejb-jar.xml“ bzw. „web.xml“, die dann die Sicherheitsmechanismen festlegen und aktivieren.
Die Einstellungen für die Security Domäne an sich gehören zur Konfiguration des Application Servers und werden nicht über die Entwicklungsumgebung vorgenommen. In der Datei „auth.conf“ im JBoss Konfigurationsverzeichnis sind die Domäne und die Sicherheitsmechanismen deklariert, beides zusammen wird auch Login Modul genannt. Bei den Login Modulen handelt es sich um fertige oder selbst geschriebene Programme, die die Authentifizierung vornehmen und je nach Bedarf auch spezielle Hardware, wie z.B. SmartCard-Lesegeräte, ansteuern können. Für die Clevery Sicherheit wurde ein Standard Login Modul gewählt, das eine Authentifizierung über eine Datenbank (mittels JDBC) abwickelt. Die Security Domäne hat die in Abbildung 8-11 dargestellte Form.
Dieses Login Modul benötigt eine Benutzernamen- und Passwort-Übereinstimmung. Dazu wird aus der Tabelle User ein Passwort (Spalte „Password“) ermittelt und mit dem eingegebenen Passwort verglichen. Bei Übereinstimmung werden dem Benutzer die Rollen zugewiesen, die in einer weiteren Tabelle, hier UserRoles, für ihn festgelegt wurden. Die Berechtigungen für diese Roles sind durch die Security Roles in den Assembler-Diagrammen festgelegt worden. Damit ist dem System bekannt welche Rechte der eingeloggte Benutzer hat auf die Komponenten zuzugreifen.

```
Clevery {
/* A JDBC based LoginModule
LoginModule options:
dsJndiName: The name of the DataSource of the database containing the Principals, Roles
              tables
principalsQuery: The prepared statement query equivalent to:
                       „select Password from Principals where PrincipalID=?“
rolesQuery: The prepared statement query equivalent to:
              „select Role, RoleGroup from Roles where PrincipalID=?“
*/
  org.jboss.security.auth.spi.DatabaseServerLoginModule required
  dsJndiName=„java:/DefaultDS“
  principalsQuery=„select Password from User where Login=?“
  rolesQuery=„select Role, RoleGroup from UserRoles where Login=? ;
};
```

Abbildung 8-11: Security Domäne

Bei der Implementierung dieses Konzeptes mit Together 6.0 traten aber Probleme während des Deployvorgangs auf. Together 6.0 erstellt insbesondere die „permissions“ für die „home methods“ falsch. Will man z.B. eine Permission für die „create“-Methode setzen, so wird stattdessen eine Permission für eine „ejbCreate“-Methode gesetzt. Zudem unterstützt Together 6.0 es nicht, eine Methode ungeprüft (unchecked) zu belassen, d.h. es müssen Permissions vergeben werden. Beide Unzulänglichkeiten zusammen genommen gestatteten es nicht Together 6.0 weiter zu verwenden.
Zur Lösung des Problems wurde dann die Erzeugung von Clevery über Ant–Skripte [ant], statt mit Together, abgewickelt. Diese Umstellung kostete ca. 3 Manntage Entwicklungsarbeit, in denen ein Skript für den Server (build.xml) und eins für den Client (build-client.xml) erstellt wurden. Die Ant-Skripte übernehmen im Prinzip alles von der

Erzeugung der Interfaces und Deploymentdeskriptoren über die Kompilierung aller Klassen hin bis zum Deployment. Im Quelltext müssen nun allerdings mit Hilfe von XDoclets [XDoc] direkt die Rechte vergeben bzw. festgelegt werden. Die zuweilen auch sehr unübersichtliche grafische Editierung der Sicherheitsparameter in Together ist durch diese Methode vollständig ersetzt worden.

8.2.4.3 Ergebnisse

Die Implementierungen der Sicherheitsmechanismen und ebenso die Sicherheitsmechanismen an sich funktionieren problemlos. Es ist auch möglich diese Mechanismen zu deaktivieren, so dass für verschiedene Tests auf die Sicherheitsrestriktionen keine Rücksicht genommen werden muss. Ein weiterer positiver Effekt ist, dass das Projekt vollständig unabhängig von der Entwicklungsumgebung geworden ist, d.h. Together kann durch jede andere IDE ersetzt werden.
Auch die Implementierung der JAAS-Funktionalität in der späten Projektphase hat sich nicht als negativ herausgestellt. Es hat sich gezeigt, dass eine Applikationssicherung durch JAAS flexibel und einfach zu realisieren ist. Zudem hat sich gezeigt, dass eine Ant + XDoclet - Lösung deutliche Vorteile mit sich bringt und aus diesem Grund am besten gleich zu Projektstart hätte eingesetzt werden sollen.

8.2.4.4 Probleme

Die meisten Probleme bereiteten der Bug in Together 6.0 und die schwierig dargestellte und missverständlich dokumentierte Materie JAAS.
Der Bug in der Together 6.0 Entwicklungsumgebung hätte, wäre er nicht rechtzeitig erkannt worden, zu erheblichen Problemen führen können, wenn der IDE zu sehr „vertraut“ worden wäre.
Weiterhin hätte der in Together 6.0 integrierte Editor für Assembler-Diagramme die Implementierung sehr erschwert. Durch die Modellierung von Permisson-Zugehörigkeiten per Assembly Links wäre eine Übersichtlichkeit über die modellierten Anwendungskomponentenzugehörigkeiten nicht mehr gegeben. Die Fehlerwahrscheinlichkeit wäre sehr hoch und die Möglichkeiten des Debuggens eklatant unkomfortabel.

8.2.5 Entwicklungsumgebung

8.2.5.1 Together

8.2.5.1.1 Beschreibung

Das Programm Together Control Center oder kurz Together der Firma Togethersoft ist eine visuelle Entwicklungsumgebung für Java, C++ und andere Sprachen.
Es ermöglicht das Modellieren mit UML-Diagrammen und erzeugt aus den Diagrammen automatisch einen lauffähigen Coderahmen. Die eigentliche Logik kann dann mit dem Together-eigenen Editor implementiert werden. Hierbei benutzt Together das Konzept der Live-Sources, d.h. dass die Diagramme automatisch an Codemodifikationen angepasst werden und umgekehrt. So muss man sich nicht mehr um die Konsistenzerhaltung zwischen Code und Modell bemühen. Das funktioniert auch, wenn der Code mit einem externen Editor verändert oder ein, komplett mit einem anderen Werkzeug erstellter Code, eingefügt wird.
Together unterstützt den Entwickler weiterhin beim Debuggen und beim Deployen von Enterprise Anwendungen und lässt sich mit vielen Plugins wie z.B. für JUnit, CVS usw. erweitern.

8.2.5.1.2 Vorgehensweise

Wir haben uns recht früh für den Einsatz von Together in der – damals aktuellen - Version 5.5 entschieden. Hierfür gab es mehrere Gründe. Zum einen gilt Together in der Wirtschaft als das zurzeit beste Modellierungswerkzeug, zum anderen hatten auch schon einige Teilnehmer der Projektgruppe Erfahrungen mit dem Programm. Die PG „Chairware“ hatte ebenfalls positive Erfahrungen mit Together gemacht. Nicht zuletzt spielte auch die Verfügbarkeit der Lizenzen am Lehrstuhl eine Rolle. Weiterhin konnte positiv bewertet werden, dass es für den freien Application Server JBoss, für den wir uns entschieden haben, ein Deploy-Plugin gab.
Ursprünglich war geplant Together nur als Modellierungswerkzeug für die Erstellung der Klassendiagramme zu nutzen und in der Implementierungsphase dann eine andere Entwicklungsumgebung (wie z.B. den JBuilder) einzusetzen.
Nachdem wir die Arbeit mit Together aufgenommen hatten stellte sich schnell heraus, dass Together unseren Entwicklungsprozess optimal unterstützt und wir haben uns deshalb entschieden die Entwicklung der Businesslogik komplett mit Together durchzuführen. Im letzten Abschnitts des Projektes haben wir uns jedoch von Together gelöst, da es doch einige Schwierigkeiten bei der Implementierung von JAAS mit sich brachte (siehe dazu Kapitel „Sicherheitskonzept JAAS“).
In der Entwurfsphase wurden die Entitätenschicht sowie die Sessionschicht definiert. Zunächst wurden alle Beans als Interface modelliert, was sich aber nicht als sinnvoll erwies, weil es keinen Weg gibt, aus den Interfaces automatisch Beans zu generieren.
In der Implementierungsphase wurden die Interfaces dann durch Session- bzw. EntityBeans ersetzt, die sich mit Together recht komfortabel anlegen lassen. Hierbei fiel zum ersten Mal auf, dass Together nicht die endgültige Version der EJB 2.0 Spezifikation erfüllte. Dieses äußerte sich darin, dass Together Code generierte, in dem (in den Remote Interfaces der Bean) EJBExceptions geworfen wurden. Dies führte dazu, dass unser Application Server (JBoss) Fehlermeldungen beim Deployen ausgab, die wenig aussagekräftig waren. Nachdem der Fehler lokalisiert wurde, war die Behebung aber leicht zu bewerkstelligen.
Da Together den Deploy-Vorgang für JBoss nicht direkt unterstützt, musste zusätzlich das Deployment-Tool von M. J. Swainston-Rainford angeschafft werden, welches sich als Plugin in Together integriert. Nebenbei korrigiert es noch einige Fehler von Together (vor allem die EJB 2.0 Spezifikation betreffend) und unterstützt den von uns benutzten MVC Persistence Manager.
Ein weiterer Fehler in Together 5.5 zwang uns dann zum Umstieg auf Together 6.0, welches zu diesem Zeitpunkt nur in einer Betaversion verfügbar war. In der Version 5.5 konnte Together keine Eins-zu-Viele-Relationen zwischen Entitäten modellieren bzw. machte dies nur fehlerhaft. In Together 6.0 funktionierte dies problemlos. Außerdem wurde der Editor stark verbessert, was die Arbeit erleichtert. Beispielsweise werden Fehler im Code rot markiert und fehlende Klassen können mit einem Druck auf Alt-Enter importiert werden.
Zum Deployen der Applikation muss ein Deployment Diagramm angelegt werden. Hierzu müssen alle Beans der Applikation sowie sämtliche lokalen Referenzen zwischen Beans eingetragen werden. Das führt dazu, dass das Diagramm schnell beliebig unübersichtlich wird.
Der eigentliche Deploy-Vorgang gestaltet sich dann mit dem oben genannten Deploy-Tool sehr einfach und wird im Wesentlichen automatisch durchgeführt. Störend hierbei fällt auf, dass Together immer den kompletten Code neu kompiliert und den Deploymentdeskriptor neu erzeugt, was den Deploy-Vorgang unnötig in die Länge zieht.
Als wir gegen Ende des Projektes JAAS einbinden wollten, was Together laut Togehtersoft auch unterstützt, stellten wir fest, dass Together die Zugriffsrechte für die Home-Methoden falsch erstellt. Da hierdurch z.B. keine Rolle das Recht bekommen kann eine Bean zu erzeugen, wird Together bei Benutzung von JAAS völlig unbrauchbar, zumindest was die Erzeugung des Deploymentdeskriptors betrifft.

8.2.5.1.3 Ergebnisse

Die einerseits sehr komfortable automatische Codegenerierung von Together ist gleichzeitig auch der größte Schwachpunkt des Programms. Leider entspricht der generierte Code an einigen Stellen nicht den Spezifikationen, was zwar sehr ärgerlich ist, aber noch leicht manuell korrigiert werden kann. Weitaus schlimmer ist die fehlende oder fehlerhafte Unterstützung von einigen Teilen des EJB Standards. Da der Deploymentdeskriptor von Together bei jedem Deploy-Vorgang neu erzeugt wird und manuelle Änderungen nicht berücksichtigt werden, kommt es zu einem inakzeptablen Zusatzaufwand. Hierdurch wird es insbesondere bei Einsatz von JAAS unmöglich das Deployment–Feature von Together zu nutzen.
Unverständlich ist an dieser Stelle die Bugfix – Politik seitens Togethersoft: Obwohl viele Fehler bereits bekannt sind, wird auf Anfrage auf die nächste Together Version verwiesen.

8.2.6 Laufzeitumgebung

8.2.6.1 JBoss als J2EE-Applicationserver

8.2.6.1.1 Beschreibung

Wie in vorangegangenen Kapiteln bereits erwähnt, war eine wesentliche Aufgabe der Projektgruppe die softwaretechnische Unterstützung der Geschäftsprozesse am Fachbereich Informatik. Dieser Teil der Zielsetzung ließ bereits darauf schließen, dass zur Erfüllung der Anforderungen komplexere Programmabläufe notwendig sein würden.
Da ein Application Server sich hervorragend dazu eignet Business Logik zu realisieren und diese nach außen hin so zu kapseln, dass sie von mehreren Clients transparent genutzt werden kann, war das erste Argument für den Einsatz eines Application Servers bereits gefunden.
Bei der Modellierung der bestehenden Geschäftsprozesse des Fachbereichs ergab sich dann frühzeitig eine weitere Anforderung an das zu realisierende System. Es stellte sich heraus dass für die sinnvolle Unterstützung der Geschäftsprozesse eine umfassende Datenhaltung notwendig sein würde, auf die von mehreren Seiten gleichzeitig zugegriffen wird. Die Realisierung dieser Anforderung sollte durch den Einsatz eines Application Servers wesentlich erleichtert werden.
Als mögliche Application Server kamen, oberflächlich betrachtet, zunächst mehrere Produkte in Betracht. Bei genauerem Hinsehen schied jedoch die Mehrzahl der Produkte aus, da sie die neuste J2EE Spezifikation nicht unterstützten, auf deren Vorteile die PG nicht verzichten wollte. Unter den verbleibenden Application Servern war dann nur JBoss [jbo] in der Version 2.4.4 frei verfügbar, so dass die Entscheidung hiermit gefallen war.
Hierbei ist jedoch zu erwähnen, dass JBoss für sich genommen noch kein vollwertiger Application Server im Sinne der J2EE Spezifikation ist. Zur Erfüllung dieser Aufgabe benötigt JBoss weitere Softwarepakete, die sich aber recht leicht mit JBoss verknüpfen lassen und zusammen die erwartete Funktionalität bieten. So wurden innerhalb der PG Tomcat [apache] als Servlet-Engine und Apache [apache] als HTTP-Server eingesetzt. Diese Pakete sind ebenfalls frei verfügbar.
Im Hinblick auf die Argumente, die für die Wahl von JBoss als Application Server gesprochen haben, ist hier zunächst zu erwähnen, das JBoss genau genommen weder frei im Sinne von „umsonst“ ist, noch die J2EE Spezifikation 2.0 vollständig unterstützt. So ist ein sinnvoller Einsatz von JBoss eigentlich nur dann möglich, wenn man einige zusätzliche Investitionen getätigt hat. Hierzu gehört beispielsweise die Dokumentation, ohne die es nicht immer möglich ist alle Fähigkeiten des Application Servers auszunutzen. Die für die PG wichtige Funktion der Container-Managed-Persistence wird von JBoss 2.4.4 auch erst nach der Installation eines speziellen Tools (siehe MVC Persistence Manager) möglich, das ebenfalls separat erworben werden muss. Alles in allem ist jedoch festzuhalten, dass die durch JBoss entstandenen Kosten weit unter den Kosten liegen, die durch den Einsatz einer kommerziellen Lösung entstehen würden. Sie beliefen sich auf etwa $ 300 für Dokumentation, weiter oben beschriebene Deploy-Tool und den Persistenzmanager.

Im laufenden Betrieb erwies sich der gewählte Application Server als relativ instabil. So kam es des Öfteren zu nicht nachvollziehbaren Abstürzen während des Deploy-Vorganges, die einen Neustart des Servers notwendig machten.

8.2.6.2 Tomcat 4.0.1

8.2.6.2.1 Beschreibung

Apache Tomcat [Tomcat] ist der Servlet Container, der die offizielle Referenzimplementierung der Java Servlet- und JavaServer Pages-Technologie darstellt. Im Rahmen der Projektgruppe kam der im JBoss 2.4.4 eingebettete Tomcat 4.0.1 zum Einsatz. Dieser implementiert die Servlet 2.3 und JSP 1.2 Spezifikationen.

8.2.6.2.2 Vorgehensweise

Der Tomcat Container wurde im Zusammenhang mit Struts eingesetzt. Durch die direkte Einbettung in den Application Server (Tomcat wird im Paket mit JBoss angeboten) fiel nur minimaler Konfigurationsaufwand an. Auch das Zusammenspiel beider Applikationen war aus diesem Grund völlig problemlos. Die Abstimmung war von vornherein als beispielhaft zu bezeichnen, was nicht zuletzt auch damit zu tun hat, dass Tomcat eine ausgesprochen große Unterstützung weltweit erfährt und sehr ausgereift ist. Mit Hilfe des Struts-Frameworks waren die dynamischen Seiten einfach zu erstellen, was ganz im Gegensatz zum Umgang mit Struts an sich stand, welches verschiedene Probleme mit sich brachte. Die einhellige Meinung über Tomcat als Container Engine war äußerst positiv und alle die mit Tomcat gearbeitet haben, würden ihn jederzeit wieder einsetzen.

8.2.6.2.3 Ergebnisse

Probleme traten bei Tomcat praktisch nicht auf. Er arbeitet stabil und zuverlässig, ohne negativ auf sich aufmerksam zu machen.

8.2.6.3 PostgreSQL 7.1.3

8.2.6.3.1 Beschreibung

Das Konzept des Projektes „Clevery“ macht auch den Einsatz einer Datenbank notwendig. Das Einsatzgebiet der Datenbank ist in diesem Projekt die persistente Haltung aller Geschäftsdaten. Eine weitere Forderung war, dass die Sicherheitsmechanismen direkt über die Datenbank Benutzer authentifizieren können.
Zwei Kandidaten kamen in Frage: mySQL und PostgreSQL [postgres]. Beiden gemein sind die hohe Produktreife und die gute Dokumentation, die durch die freie Verfügbarkeit und die große Verbreitung der beiden Systeme begründet ist. Die Voraussetzung für die Zusammenarbeit mit Clevery war lediglich die JDBC Schnittstelle, über die beide Kandidaten verfügten. Die Wahl des DBMS fiel aus folgenden Gründen auf PostgreSQL:

- PostgresSQL ist frei verfügbar und wird in der von uns verwendeten Linux-Distribution auf Wunsch mit installiert.
- Die Datenbank ist transaktionssicher (entscheidendes Argument).
- Eine Anbindung an JBoss ist in seiner Distribution schon vorkonfiguriert und wurde in diversen Foren als problemlos und empfehlenswert beschrieben.

8.2.6.3.2 Vorgehensweise

Um die Container-Managed-Persistens (CMP) nach der letzten Spezifikation von J2EE nutzen zu können, musste Clevery mit Datenbankunterstützung ausgestattet werden. Die eigentliche Umsetzung der J2EE Spezifikation macht es leider nötig, einen externen CMP Manager einzusetzen, da der JBoss Server in der eingesetzten Version in diesem Punkt noch Mängel aufwies. Die generelle Anbindung der Datenbank erfolgte über eine JDBC-

Schnittstelle [jdbc], so dass die Wahl des Datenbankmanagementsystems (DBMS) weder durch den CMP Manager noch durch den Application Server an sich beeinträchtigt wurde.
Der Aufwand PostgreSQL an den JBoss anzubinden war an sich minimal. In den JBoss-Konfigurationsdateien mussten nur wenige Parameter angepasst werden. Die Module mit den JDBC-Schnittstellen mussten ebenfalls zu der JBoss Konfiguration hinzugefügt werden. Damit war die Anbindung abgeschlossen und die Datenbank arbeitete stabil und fehlerfrei.
Den Sicherheitsanforderungen trug die Datenbank ebenfalls durch fertige Login-Module, die nur noch formal auf die Tabellenstruktur der Datenbank von Clevery zugeschnitten werden mussten, Rechnung. Auch an dieser Stellt war kein Zugriff auf die API des DBMS notwendig, da wiederum der Zugriff über JDBC erfolgte.
In den frühen Phasen der Implementierung von Clevery, als noch nicht auf der zentralen Datenbank des (Integrations-)Servers gearbeitet wurde, existierte auf jedem Arbeitsplatzrechner eine eigene Datenbank. Um die ausschließlich für UNIX ausgelegte PostgreSQL Datenbank unter Windows 2000 betreiben zu können, musste zusätzlich ein UNIX-Layer installiert werden. Dieser UNIX-Layer wurde durch „cygwin“ [cygwin] implementiert und bietet dem Anwender als Eingabeschnittstelle eine typische Unix-Benutzershell. Es war auf der einen Seite unkomfortabel immer erst cygwin und dann PostgreSQL starten zu müssen, bevor auf die Datenbank zugegriffen werden konnte. Auf der anderen Seite stand dann aber das volle Funktionsspektrum des DBMS zur Verfügung. Ein Pluspunkt für cygwin war auch, dass es eine fertig kompilierte Version von PostgreSQL gab, die direkt mitinstalliert werden konnte. Eine Übersetzung des Quellcodes war nicht notwendig.

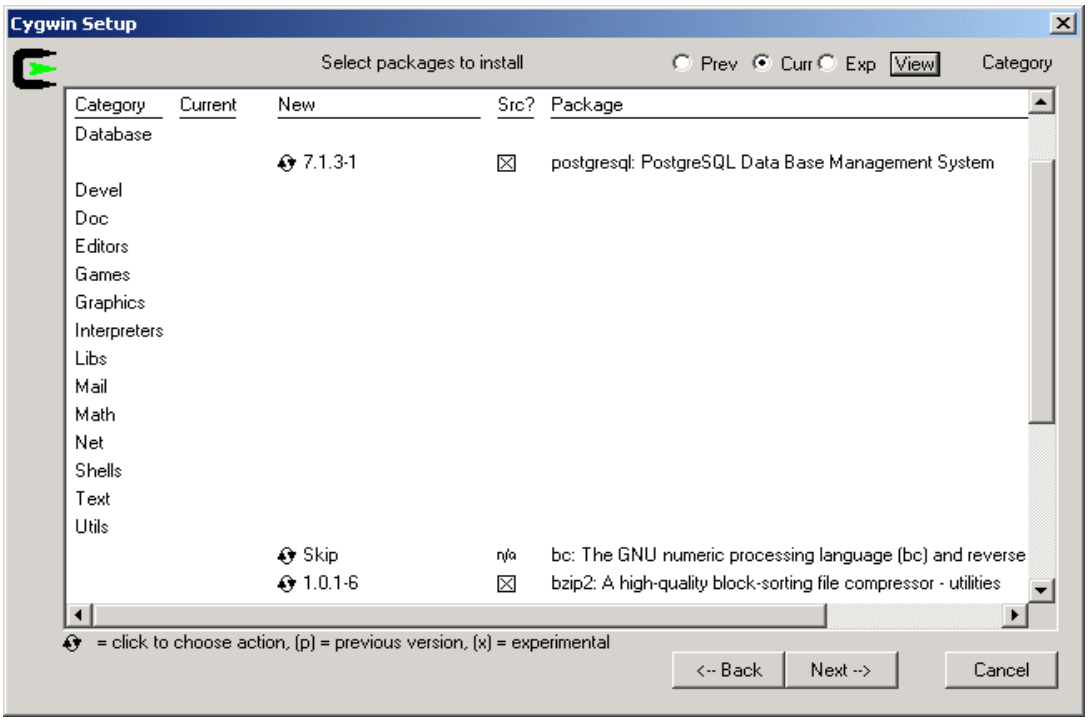


Abbildung 8-12: cygwin Setup

Nur für die Interprozesskommunikation zwischen Windows und cygwin musste noch das Paket „cygipc-1.11-1.tar.bz2“ [cygwin] zusätzlich von Hand nachinstalliert werden, da es nicht zur Standardinstallation gehört. Dieses Paket enthält einen IPC-Daemon, der die Interprozesskommunikation abwickelt. Dieser muss vor dem DBMS unter cygwin gestartet werden.
Die Verfügbarkeit der Datenquellen ist auch eine wichtige Startvoraussetzung für den Application Server JBoss, der diese beim Starten einbindet. Das bedeutet, dass vor dem JBoss das DBMS gestartet werden muss. Im Gegensatz zum JBoss reichte es fast immer

aus, die Datenbank einmal zu Beginn einer Arbeitssitzung zu starten. Sie lief dann fast immer anstandslos, während der Application Server häufig neu gestartet werden musste.
Das Administrationstool „pgAdmin II“ [pgadm], das für die Verwaltung des DBMS sowie für das Erzeugen von Datenbankobjekten eingesetzt wurde, war von der Bedienung her nicht ausgereift. Es gab, wenngleich selten, Fehlermeldungen, die jedoch zu keiner Beeinträchtigung der Funktionalität führten und unter Umständen auch im Zusammenhang mit dem Einsatz von einem UNIX-Layer unter Windows 2000 stehen.

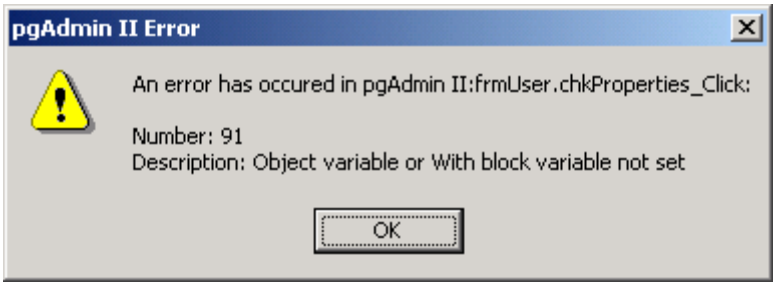


Abbildung 8-13: Fehler ohne Folgen

Das Programm „pgAdmin II“ ersparte die Einarbeitung in die Kommandozeilenbefehle von PostgreSQL und die teilweise recht mühsame Anwendung von SQL-Statements. Die Verwendung war recht intuitiv, so dass die Pflege der Datenobjekte keinen nennenswerten Aufwand mit sich brachte.
Für einen Basisdatenbestand, der zu Testzwecken immer vorausgesetzt werden konnte, wurden Skripte in SQL angelegt. Mit deren Hilfe konnten die Datenbanken immer auf einen einheitlichen Stand gebracht werden.

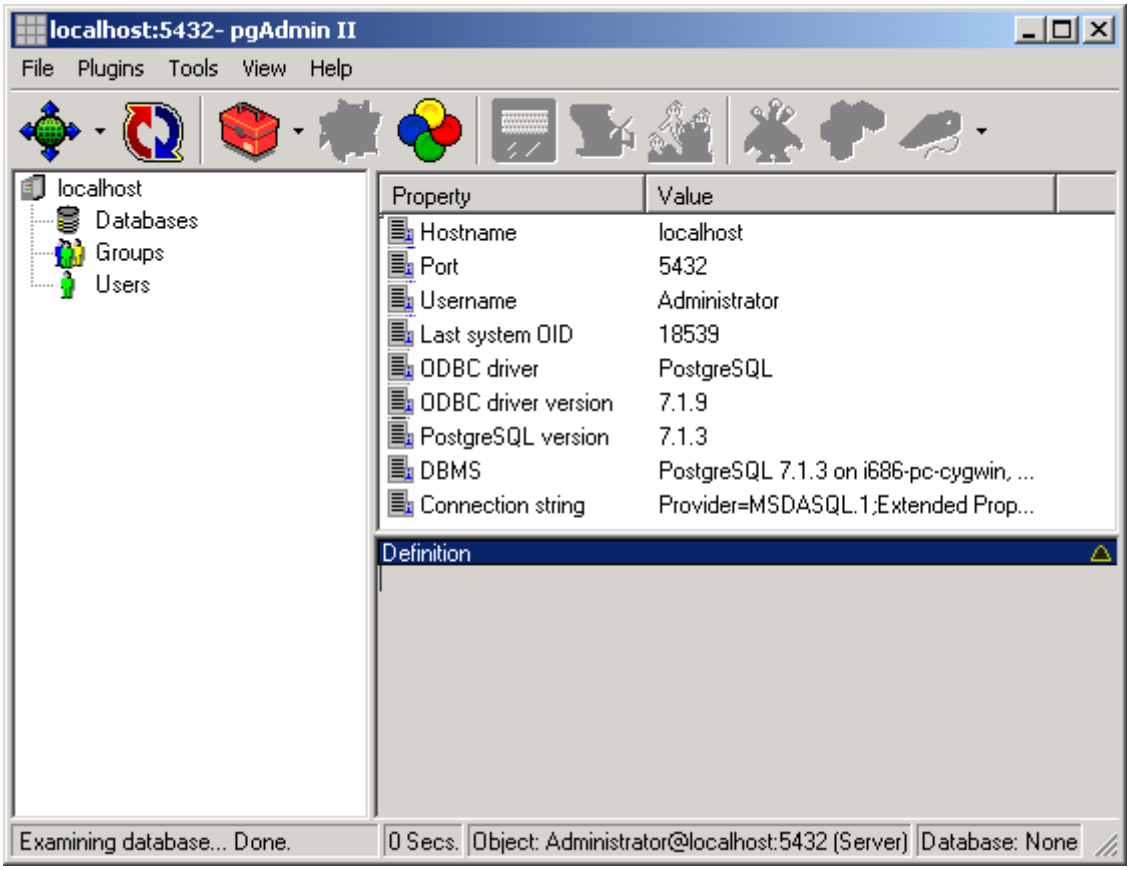


Abbildung 8-14: pgAdmin II Oberfläche

8.2.6.3.3 Ergebnisse

Es gab nur ein echtes Problem mit dem Laufzeitverhalten von PostgreSQL: Oft wurden nach dem teils unsachgemäßen Beenden von PostgreSQL nicht alle angelegten temporären Dateien entfernt bzw. freigegeben. PostgreSQL versagte dann beim nächsten Start den Dienst mit einem Hinweis auf diese Dateien. Mit dem Löschen der Datei „postmaster.pid“, die in der Fehlermeldung als Ursache genannt wurde, wurde dieses Problem gelöst.

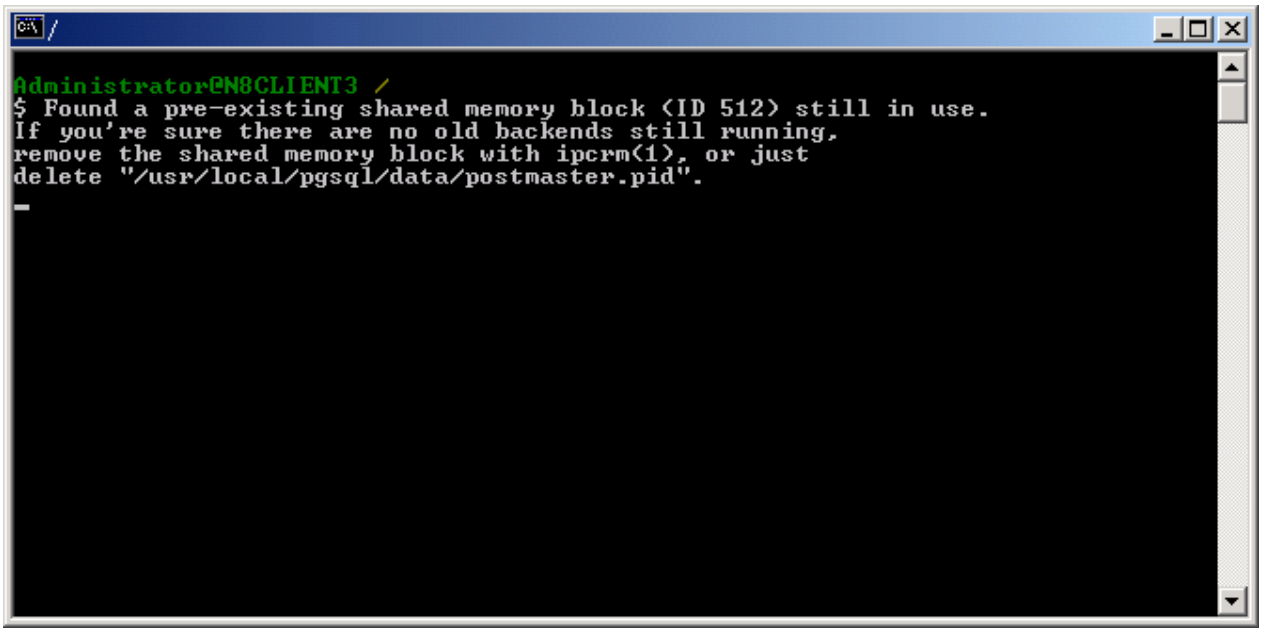


Abbildung 8-15: Probleme nach unsauberem Abbruch

Weitere Probleme gab es kaum, jedoch war das Arbeiten mit den Datenbankinstallationen auf den einzelnen Arbeitsplatzrechnern bei der Betrachtung des Startverhaltens sehr unpraktikabel. Der Startablauf sah wie folgt aus: Einloggen, cygwin starten, IPC-Daemon starten, PostgreSQL starten, JBoss starten. Das ist nicht sehr anwenderfreundlich. Natürlich wurde cygwin auch als Systemdienst konfiguriert und das Starten des DBMS per Skript automatisiert. Wegen der Probleme beim Herunterfahren des DBMS wurde letztendlich nur noch der IPC-Daemon „cygipc“ als Systemdienst gestartet. Ein Startskript für cygwin steuerte dann den Startvorgang des DBMS und beugte der oben geschilderten Problematik vor, indem es generell die Datei „postmaster.pid“ vor dem Start von PostgreSQL löschte. Diese Problematik erübrigt sich durch die Nutzung eines zentralen DBMS auf dem Linux-Server.
Die Dokumentation zu PostgreSQL an sich war vorbildlich, die für die Integration in eine Enterprise-Umgebung weniger. Es gab nicht viele Parameter, die angepasst werden mussten, doch bis eine funktionierende Konfiguration stand, war sehr viel Recherche- und Probieraufwand nötig.
Ein organisatorisches Problem war, dass die Datensätze der Datenbanken auf dem Arbeitsrechnern nicht konsistent gehalten wurden. Durch die flexible Nutzung der Arbeitsplätze kam es häufig vor, dass die lokale Datenbank anders aussah als die, die man vorher auf einem anderen Rechner benutzt hatte.
In Anbetracht der vielen Tools, die nötig waren, um die Datenbank unter Windows zu betreiben, sollte ein Einsatz von PostgreSQL unter Windows gründlich überdacht werden, denn die JDBC – Schnittstellen bieten auch andere Systeme, die zudem als native Pakete für Windows verfügbar sind. Als zentrales DBMS ist PostgreSQL in einer Unix-Umgebung aber in jedem Fall eine brauchbare Variante.

8.3 Test und Abnahme

8.3.1 Test Umgebungen

Als Testumgebung für Clevery wurden keine standardisierten Testverfahren eingesetzt. Aufgrund der ständigen Weiterentwicklung entschloss man sich gegen die Implementation festgelegter Integrationstests.
Stattdessen wurde für den Test der Geschäftslogik auf ein selbst entworfenes Tool gesetzt, dass eine dynamische Weboberfläche zum Test der Geschäftslogik bietet. Somit musste das Testtool mit der Weiterentwicklung nicht verändert werden.
Das Testtool besteht aus einem Servlet, welches man zusammen mit der Applikationslogik deployen kann. Beim Aufruf des Servlets im Browser werden per Java Reflection API alle Geschäftsmethoden einer zuvor ausgewählten Session-Bean aufgelistet. Danach wird anhand der Methodensignatur und anhand des Quelltextes der Methode ein Eingabeformular mit einem Eingabefeld für jeden Parameter der Methode erzeugt. Der Ablauf eines Tests sieht im Detail dann so aus:

1. Auswahl eines Business Service Controllers (Session-Bean)
2. Auswahl einer Business Methode dieses Controllers
3. Eingabe der Aufrufparameter dieser Methode in ein dynamisch generiertes Eingabeformular.
4. Abschicken dieses Formulars und damit Aufruf der Methode.
5. Ausgabe eines Objektbaums, den die Methode als Rückgabe hat.

Für einen derart simplen Test müssen allerdings ein paar Vorraussetzungen erfüllt sein, an die wir uns beim Design der Anwendung gehalten haben:

1. Das Design Pattern Session Facade musste konsequent eingehalten werden, um alle Geschäftsprozesse über Session Beans zugänglich zu machen.
2. Die Geschäftsmethoden sollten möglichst elementar und von einander unabhängig sein, um sie einzeln testen zu können, da das Testtool nur jeweils eine Methode pro Aufruf ausführen kann.
3. Alle Parameter einer Session Bean sollten einfache Datentypen sein, um für sie eine einfache Eingabemöglichkeit per Formular zu schaffen. Für die Parameterübergabe von Kollektionen von einfachen Datentypen wurde ein Eingabeschema in Form eines String Pattern geschaffen.
4. Der Rückgabewert einer Methode sollte eine möglichst anschauliche Darstellung als Text mit Hilfe einer toString()-Methode bieten. Da alle Methoden der Geschäftsmethoden bei Clevery ausschließlich DynaBeans oder Kollektionen hiervon zurückgeben, wurde in der DynaBean-Implementation eine toString()-Methode kodiert, die den gesamten übermittelten Objektbaum übersichtlich ausgibt.

8.3.2 Abnahme

Für Clevery wurden regelmäßig durch den Projektleiter und den Qualitätsmanager manuelle Integrationstest der Geschäftslogik in der Benutzeroberfläche oder soweit noch nicht vorhanden mit dem oben beschriebenen Testtool durchgeführt.
Fehler und Verbesserungen wurden von Ihnen sowohl auch von allen anderen Projektteilnehmern in einem für die Projektgruppe aufgesetzten System namens Wiki eingetragen und vom Verantwortlichen abgearbeitet.

9 Clevery Anwendung

9.1 Anforderungen

9.1.1 Hardware

Die Projektgruppe bekam vom Lehrstuhl X der Universität Dortmund einen eigenen Rechnerraum inklusive der zur Entwicklung notwendigen Hardware zur Verfügung gestellt, zu dem ausschließlich die Mitglieder der PG Zugang hatten. Da das benötigte Entwicklungsszenario recht umfangreich war, konnte man es schlecht in einen öffentlichen Rechnerpool auslagern. Außerdem brachte diese Anordnung den Vorteil, ein einmal aufgebautes Entwicklungssystem weitgehend stabil halten zu können, da es vor Veränderungen durch Außenstehende geschützt war.

Die zur Entwicklung benötigte Hardware wurde eigens für die Projektgruppe bestellt. Leider kamen die einzelnen Komponenten erst nach einer mehrwöchigen Verspätung, was zu erheblichen Verzögerungen im Zeitplan führte.

Besonderes Augenmerk galt dem Server. Dieser ging aus verschiedenen Gründen als Erstes in Betrieb. Dabei kam folgende Hardware zum Einsatz:

- Athlon XP 1600+
- 512 MB DDR-RAM
- 40GB Festplatte (7200er IDE)

Der Server verfügte über keinerlei Peripherie. Es kamen weder Tastatur und Maus noch Monitor zum Einsatz, da der Platz in dem kleinen Raum sehr begrenzt war. Ein defektes Speichermodul des Servers konnte über einen Zeitraum von mehreren Monaten nicht ersetzt werden, so dass der Server bis zum Ende der PG mit nur 256 MB Hauptspeicher auskommen musste.

Bei den Workstations kamen einfachere Geräte zum Einsatz, lediglich beim Hauptspeicher konnte im Hinblick auf die zu nutzende Entwicklungssoftware kein Kompromiss eingegangen werden. Uns standen insgesamt 4 nahezu baugleiche Workstations mit folgender Hardware zur Verfügung:

- Athlon C, 1000 MHz
- 512 MB SD-RAM
- 30GB Festplatte (5400er IDE)

Alle Rechner standen in unserem PG-Raum und waren mit jeweils 100 MBit/s über einen Switch an das Universitätsnetz angebunden. Außerdem war an einer der Workstations ein Laserdrucker vom Typ HP-Laserjet 4 angeschlossen. Diese Workstation verfügte auch über einen CD-Brenner.

9.1.2 Software

9.1.2.1 Betriebsumgebung

Wir legten großen Wert darauf möglichst freie Software zu verwenden. Speziell für den Betrieb von Clevery sind folgende Softwarekomponenten zu verwenden:

- Java 2 Standard Edition ab 1.3 (PG nutzte Java 1.3.1_03)
- PostgreSQL 7.1.3 (PG nutzte PostgreSQL 7.1.3-11, unter Windows kann PosgreSQL nur in Verbindung mit Cygwin genutzt werden)
- JBoss 2.4.4 (mit integriertem Tomcat bzw. Catalina, wie die Software seit Anfang 2002 heißt)

- MVCSoft Persistence Manager 1.0 (für den Betrieb ist nur die frei erhältliche Runtime erforderlich)

9.1.2.2 Entwicklungsumgebung

Für die Entwicklung selbst wurden weitere, bis auf Together Control Center 6, ebenfalls freie Programme verwendet:

- Together Control Center 6.0 (anfangs Together Control Center 5.5)
- X-Doclet 1.1.2
- Jakarta Struts 1.1
- Jakarta Ant 1.4.1
- Jakarta Log4J 1.2.3
- LogFactor5

9.2 Installation und Konfiguration

9.2.1.1 Installation

9.2.1.1.1 Java 1.3.1

Das JDK mit der Installationsroutine installieren und eine globale Umgebungsvariable JAVA_HOME auf das Javaverzeichnis zeigen lassen.

9.2.1.1.2 PostgreSQL 7.1.3

PostgreSQL ebenfalls mit der Installationsroutine installieren. Eine weitere Konfiguration ist nicht nötig, allerdings muss eine Datenbank mit dem Namen UPDB unter dem User „postgres“ angelegt werden. Initial erforderliche Daten können per SQL-Skript eingespielt werden.

9.2.1.1.3 JBoss 2.4.4

Der Application Server lässt sich durch Entpacken der gelieferten ZIP Dateien installieren. Nach dem Entpacken ist der Server einsatzbereit, für Clevery müssen allerdings noch einige Konfigurationsdateien angepasst werden um PostgreSQL einzubinden und die richtigen E-Maileinstellungen festzulegen, außerdem ist die Erweiterung um folgende Komponenten erforderlich:

- JDBC Treiber für PostgreSQL
- MVCSoft Persistance Manager

Weiterhin ist in unserem speziellen Fall einen Hook zu unserem Apache Webserver eingerichtet. Auf diese Art wird der normale Inhalt der PG Homepage über den deutlich schnelleren Apache ausgeliefert, während alle Zugriffe auf Clevery für den Benutzer völlig transparent an den im JBoss integrierten Catalina weitergeleitet werden.

Die von uns verwendeten Konfigurationen aller beteiligten Programme liegen auf CD bei. Speziell die „JBoss 2.4.4 nightshift Edition“ sollte unverändert übernommen werden, wodurch oben erwähnte Konfigurationsschritte überflüssig werden.

9.2.1.2 Erstellung der Archive

Trotz anfänglicher Versuche die Applikation grundsätzlich mit Together Control Center und dem dafür vorgesehenen Deploy Tool zu deployen, mussten wir schließlich auf die

Verwendung von Jakarta Ant und XDoclet umsteigen. Zwei Ant-Skripte übernehmen das Compilieren, das dynamische Anlegen der Interface-Klassen und Deployment Deskriptoren sowie den nötigen Deployvorgang. Das Skript für den Server kann auch ein für die Entwicklung dringend empfohlenes Javadoc generieren. Die beiden Ant-Skripte liegen auf CD bei.

9.2.1.3 Starten der Anwendung

Zunächst muss die Datenbank gestartet werden. Erst danach darf der Application Server hochgefahren werden, da dieser ohne Datenbankverbindung nicht richtig starten kann. Wenn die Applikation noch nicht deployed wurde, muss dies als letzter Schritt erfolgen. Hierzu werden einfach das EAR und das WAR Archiv in das Deploy-Verzeichnis des Application Servers kopiert.

9.3 Anwendungsbeispiel: Anmeldung zu einer mündlichen Prüfung

Am Beispiel einer Anmeldung zu einer mündlichen Prüfung wird im Folgenden exemplarisch demonstriert, wie die einzelnen Nutzer des Systems miteinander interagieren.

9.3.1 Die Bekanntgabe eines Prüfungstermins

Zunächst muss der Prüfer seine Prüfungstermine festlegen und ins System eintragen.

Abbildung 9-1: Der Prüfer hat sich eingeloggt

In Abbildung 9-1 hat sich der Benutzer im System eingeloggt. Mit der automatisch zugewiesenen Rolle „Prüfer“ besbitzt er die Rechte Prüfungen bekannt zu geben.

Abbildung 9-2: Der Prüfer hat den Bereich „Prüfungsmanagement“ ausgewählt

Nach der Auswahl des Bereiches, sieht der Prüfer seine Standard-Rolle in der Drop-Down-Box und darunter die zugehörigen Aktionen, die er in dieser Rolle ausführen kann.
Unter dem Punkt Termine wird nun die Aktion „Mündliche Prüfung anlegen“ ausgewählt.

Abbildung 9-3: Der Prüfer hat seine Termindaten eingetragen

Dieser Punkt führt direkt zu der Eingabemaske (Abbildung 9-3), in dem der Prüfungstermin ins System eintragen werden kann. Nach der Eingabe der Daten zum Termin kann optional eine Bemerkung eintragen werden, um eine kurze zusätzliche Information an die Studenten weiterzugeben.
Nach dem Abschicken der Termindaten, ist der Termin sofort zur Anmeldung freigegeben. Studenten können diesen sowohl als Gast als auch als eingeloggter Benutzer einsehen.

9.3.2 Anmeldungsanforderung des Studenten

Wenn sich ein Student einloggt, stehen seine Daten, die er üblicherweise auf jedem Anmeldeformular immer wieder neu eintragen müsste, bereit. Das heißt, dass er nur noch die wesentlichen Daten zu einer Prüfungsanmeldung eintragen muss. Alles andere wird mit Hilfe interner Relationen bereitgestellt.
Bei der Optimierung des Anmeldungsprozesses ist uns aufgefallen, dass wir zu der eigentlichen Anmeldung gleichzeitig eine Übersicht präsentieren können, welche Prüfer welche Prüfungsgebiete prüfen, so dass sich der Student sich zunächst für ein oder mehrere Prüfungsgebiete entscheiden muss.
Anhand der ausgewählten Gebiete, ist es dem System möglich eine Auswahl von Prüfern anzuzeigen, die mindestens eines der Gebiete prüfen. Zusätzlich wird angezeigt, wie die ausgewählte Fachkombination überdeckt wird.

Abbildung 9-4: Der Student wählt einen Prüfer

Nach der Prüferauswahl, kann nun eingesehen werden, ob der betreffende Prüfer freie Termine zur Verfügung gestellt hat (Abbildung 9-5). Diese Termine entsprechen denen, die wir im ersten Schritt angelegt haben.

Abbildung 9-5: Der Student wählt einen Prüfungstermin aus

Bei näherem Interesse, kann sich der Student zu diesem Termin durch einen Klick auf den Link „Anmelden“ seine Anmeldung ankündigen. In diesem Fall öffnet sich ein weiteres Formular mit der Aufforderung die restlichen Daten auszufüllen und die Prüfungsanmeldung an den Prüfer und das ZPA weiterzureichen.

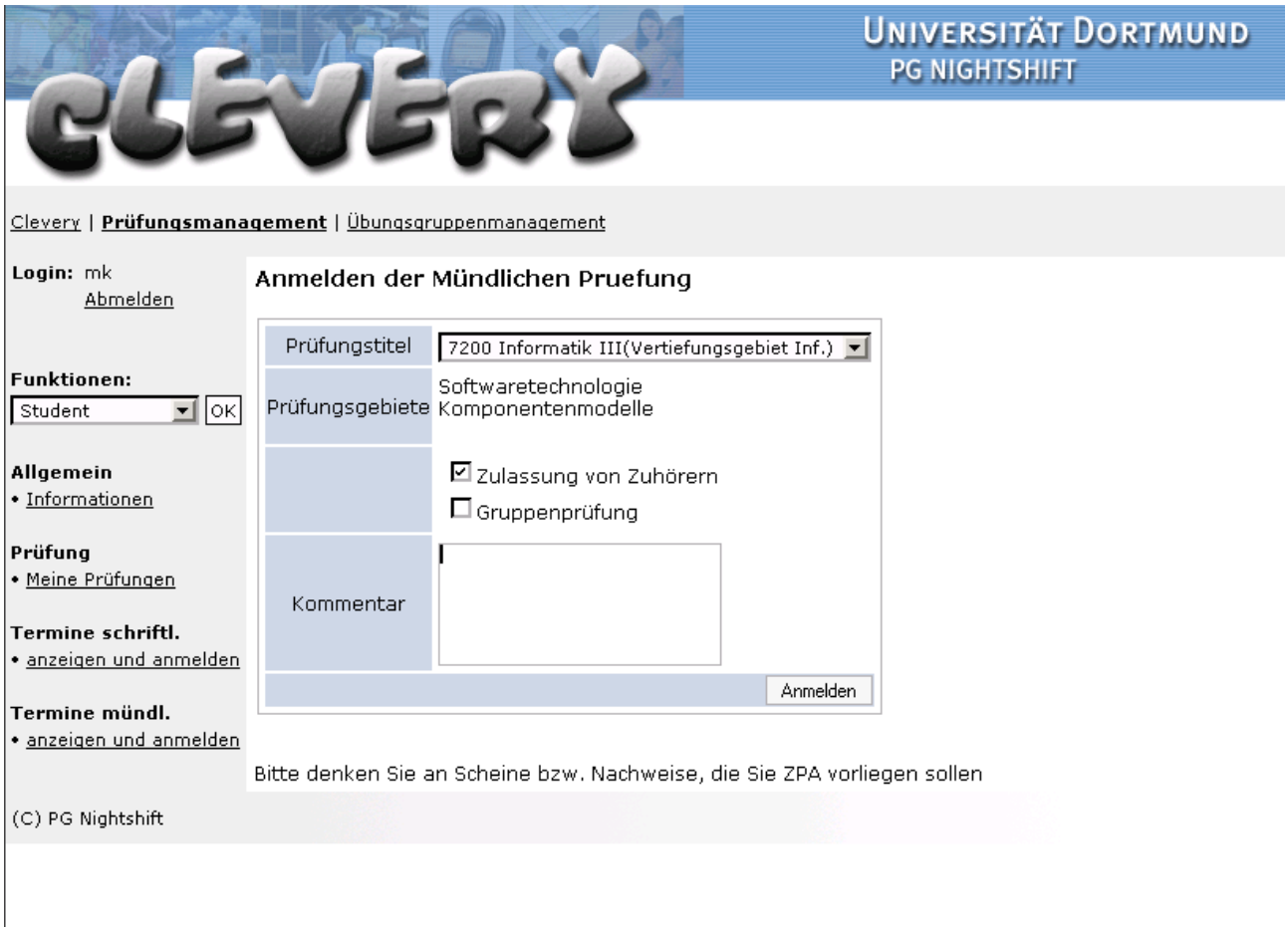


Abbildung 9-6: Die restlichen Daten der Anmeldung

Nach dem Abschicken der Anforderung der Anmeldung zur mündlichen Prüfung, kann der Student den Status seiner Anmeldung jederzeit überwachen und sich bei Bedarf wieder abmelden (oder genauer: eine Abmeldung anfordern) (Abbildung 9-7).

Abbildung 9-7: Der Student kann jederzeit den Bearbeitungsstatus seiner Anmeldung überwachen

Nebenbei sei angemerkt, dass hier auch die Möglichkeit besteht Leistungen vergangener Prüfungen einzusehen. So entsteht hier eine kleine Zusammenfassung der Laufbahn eines Studenten.

9.3.3 Die Bestätigung der Prüfungsanmeldung seitens des Prüfers

Der Prüfer kann unter seinem Login jederzeit neue Anmeldungen einsehen. Außerdem wird er bei Eingang neuer Anmeldungen per E-Mail benachrichtigt.

Abbildung 9-8: Der Prüfer sieht die neu eingegangene Prüfungsanmeldung

Der Prüfer kann nun die Anmeldung bestätigen oder auch ablehnen. In unserem Beispiel wird die Anmeldung angenommen.

9.3.4 Die Bestätigung der Anmeldung beim zentralen Prüfungsamt

Das ZPA wird durch das System automatisch per E-Mail benachrichtigt, wenn neue Anmeldungen vorliegen. Zusätzlich können die aktuellen Anmeldungen unter dem Login eines ZPA-Mitarbeiters eingesehen werden.

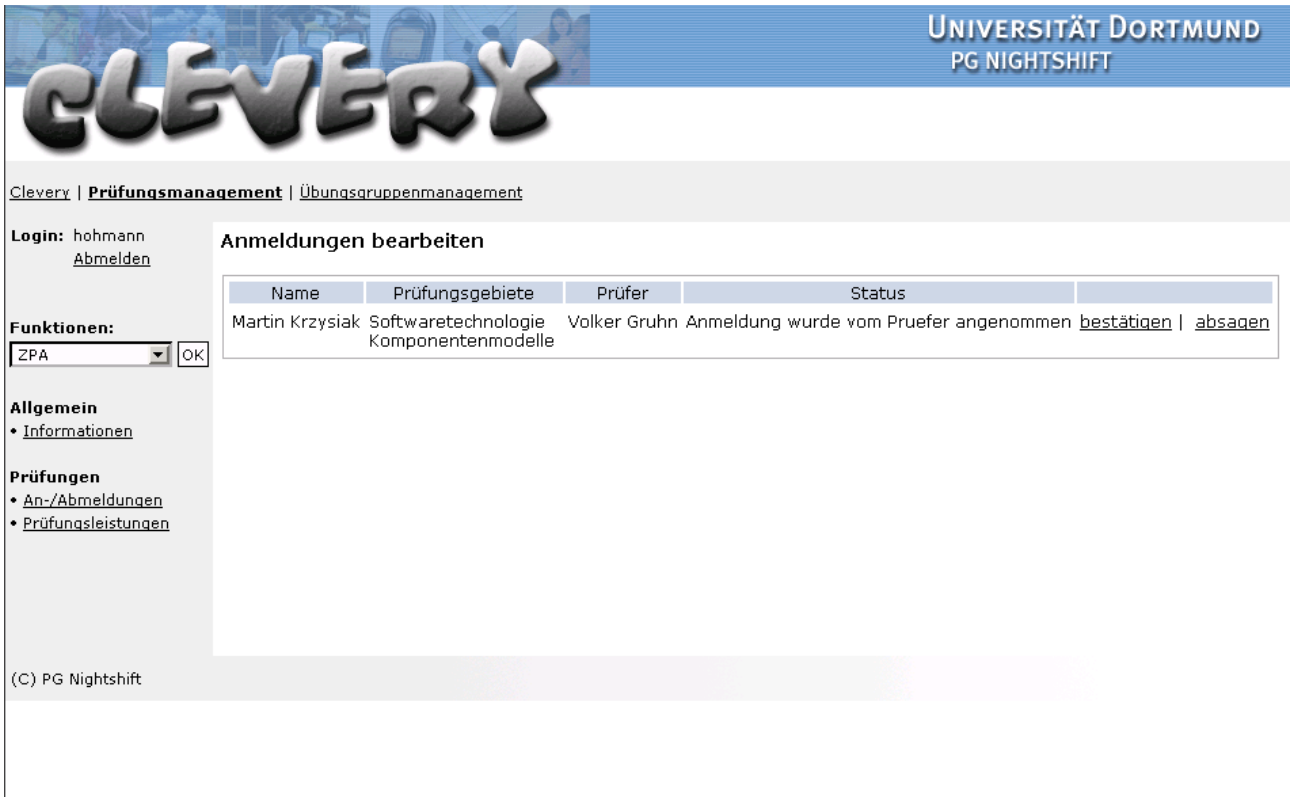


Abbildung 9-9: Das ZPA kann neu eingegangene Prüfungsanmeldungen einsehen

Nach dem Einloggen kann der Benutzer mit der Rolle ZPA unter dem Punkt „An-/Abmeldungen“, unsere neu eingegangene Prüfungsanmeldung sehen, diese überprüfen und bestätigen oder ablehnen.

9.3.5 Prüfungsvorbereitung und die Prüfung

An diesen Prozessen ist unser System nicht beteiligt. Wichtig ist es jedoch anzumerken, dass falls sich ein Student dazu entscheidet, die Prüfungsanmeldung rückgängig zu machen, kann er dieses unter seinem Login veranlassen. Der Status seiner Anmeldung steht immer unter dem Menüpunkt „Meine Prüfungen“ zur Verfügung (Abbildung 9-10).

Abbildung 9-10: Der Student kann jederzeit den Status seiner Prüfungsanmeldung einsehen und sich gegebenenfalls abmelden.

9.3.6 Bewertung der Prüfung

Nach der Prüfung obliegt es dem Prüfer die Note sowie das Protokoll in das System einzutragen. Das dafür vorgesehene Formular ist in Abbildung 9-11 abgebildet. Nach dem Ausfüllen des Formulars werden der betreffende Student und das ZPA über die eingetragene Prüfungsleistung benachrichtigt.

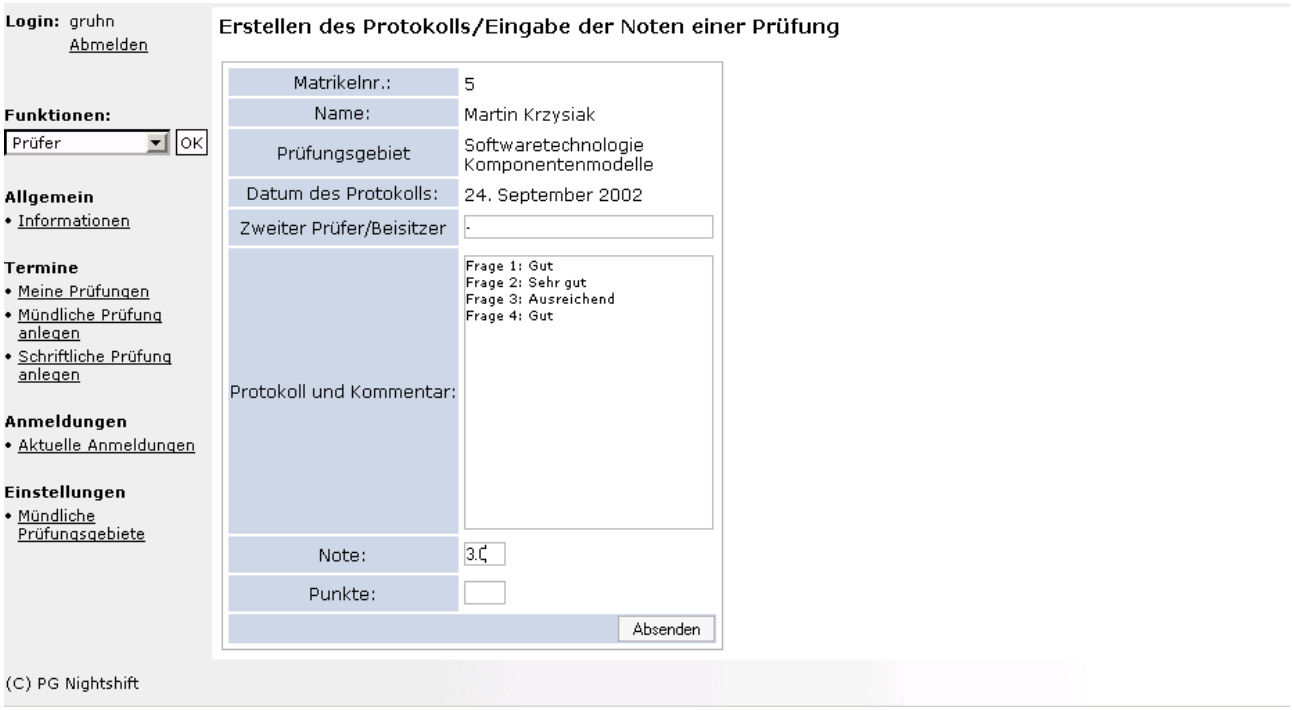


Abbildung 9-11: Der Prüfer bewertet die Prüfung.

Die Prüfungsleistung verbleibt bis zum Abschluss des Studiums protokolliert auf der „Meine Prüfungen“ Seite. Es baut sich dort eine kleine tabellarische Übersicht über die Leistungen auf.

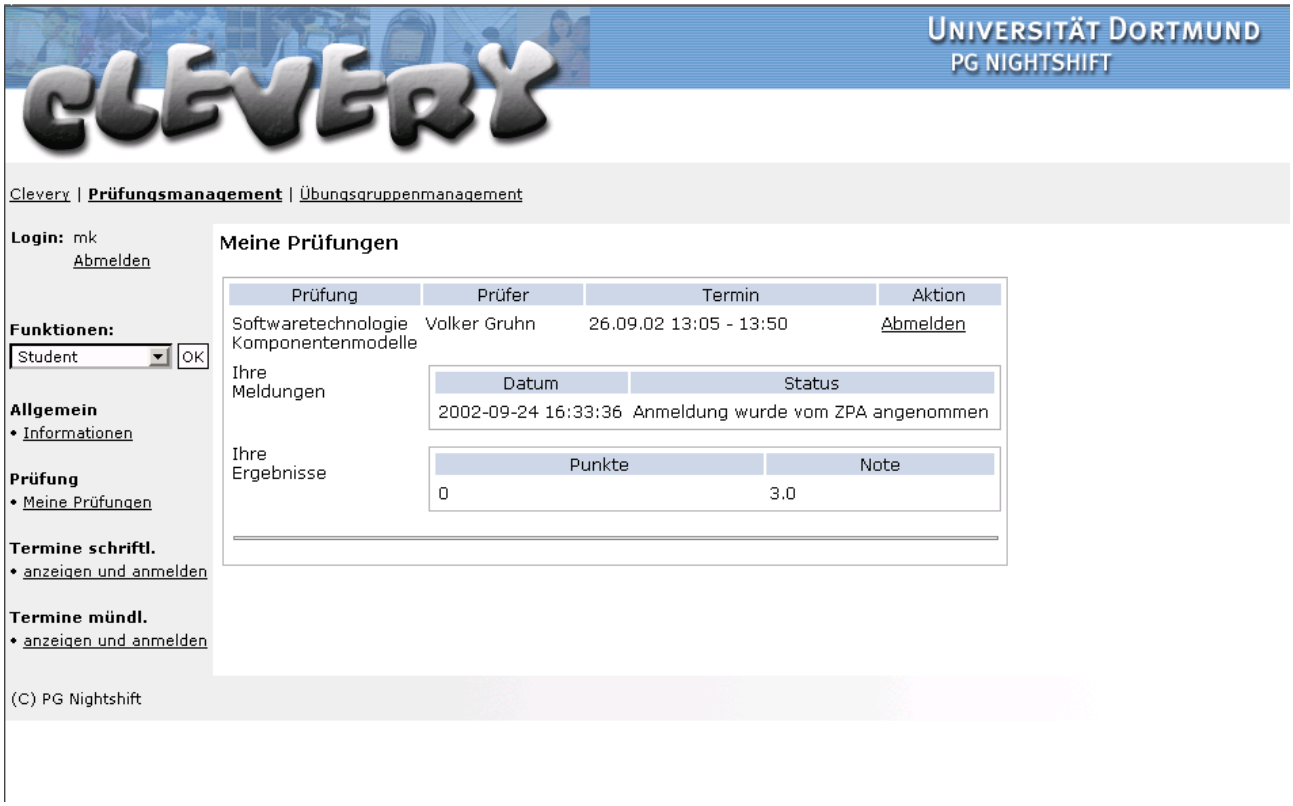


Abbildung 9-12: Der Student hat eine eigene Übersicht über seine studentische Laufbahn

Auch das ZPA ist in der Lage, jederzeit auf den aktuellen Stand der Prüfungsleistungen eines Studenten zuzugreifen und eine Übersicht anzuzeigen.
Hiermit ist der Prozess für alle Beteiligten abgeschlossen.

10 Bewertung

Im Folgenden sollen die Ergebnisse der PG bewertet werden. Hierzu wird zum einen untersucht inwiefern es der PG gelungen ist die ihr gestellten Minimalziele zu erreichen und zum anderen werden Leistungen aufgeführt, die über das Erfüllen dieser Minimalziele hinaus gehen.

10.1 Minimalziele

Unter Berücksichtigung des PG – Titels lassen sich im Wesentlichen zwei Minimalziele ableiten:

1. Dokumentation ausgesuchter Prozesse
2. Softwaretechnische Umsetzung eines Teils dieser Prozesse

Um den zweiten Punkt etwas detaillierter zu betrachten, wurde dieser in drei wesentliche Teile der Softwareentwicklung unterteilt: Anforderungsanalyse, Entwurf und Implementierung (vgl. hierzu auch Kapitel 2.6).

10.1.1 Dokumentation ausgesuchter Prozesse im Fachbereich

Legt man die Gesamtheit aller Geschäftsprozesse am Fachbereich Informatik zu Grunde, dann ist es der PG sicherlich nicht gelungen, diese vollständig zu dokumentieren. Beschränkt man sich jedoch auf die zwei wesentlichen Prozesse „Übungsgruppenmanagement“ und „Prüfungsmanagement“, dann ist die Dokumentation als vollständig zu bewerten.
Die Beschränkung auf wenige ausgesuchte Prozesse war offensichtlich notwendig, da die Dokumentation aller Geschäftsprozesse am Fachbereich den Rahmen der PG überschritten hätte und sich zudem nicht alle Prozesse softwaretechnisch unterstützen lassen.
Die Wahl fiel deshalb auf zwei zentrale Prozesse am Fachbereich, für die die PG-Teilnehmer ein grundlegendes Verständnis mitbrachten, da jeder einen Teil dieser Prozesse schon einmal durchlaufen hatte. Ein weiterer Grund für die Wahl genau dieser Prozesse ist die Tatsache, dass eine Optimierung dieser besonders geeignet schien, die Dauer des Informatikstudiums zu verringern, womit eine weitere Zielsetzung der PG abgedeckt wird.
Durch die Verwendung von LeuSmart Diagrammen in Verbindung mit beschreibendem Text verliert der Leser der erstellten Dokumentation nie den Überblick über den Gesamtzusammenhang der einzelnen Prozessteile und kann sich bei Bedarf detaillierter über die Abläufe informieren.
Weiterhin war der Systemimplementierung zur Unterstützung der Prozesse eine Optimierung der Abläufe vorangestellt. Hierzu erwies sich die erstellte Dokumentation als sehr geeignet, weil die Diagrammform ineffiziente Abläufe in der Prozesskette gut sichtbar machte.
Somit kann man abschließend sagen, dass das Ziel, ausgesuchte Prozesse am Fachbereich verständlich und übersichtlich zu dokumentieren, erreicht wurde.

10.1.2 Anforderungsanalyse eines Systems zur Unterstützung der Prozesse

Mit der Dokumentation und Optimierung der am Fachbereich bestehenden Geschäftsprozesse war bereits der erste Schritt für das Erfassen der Anforderungen getan. Auf dieser Basis wurde ein gesondertes Anforderungsdokument erstellt, das in die drei großen Bereiche Prüfungsmanagement, Übungsgruppenmanagement und allgemeine Anforderungen unterteilt wurde. Innerhalb dieser Teilbereiche wurden die Anforderungen durch eine Unterteilung in zeitliche Phasen weiter hierarchisiert, so dass eine größtmögliche Übersicht über die einzelnen Anforderungen gegeben war.
Diese Unterteilung des Anforderungsdokuments erwies sich als sehr vorteilhaft, um einzelne Systemfunktionen zu identifizieren und diese dann nach und nach umzusetzen.
Das Vorgehen beim Erstellen des Anforderungsdokuments zeigte sich jedoch in Teilen als unzureichend. So wurden Teile der notwendigen Systemfunktionalität zunächst gar nicht

erfasst, weil sie weder direkt zum Prüfungsmanagement noch zum Übungsgruppenmanagement gehörten oder aus der erstellten Dokumentation der Ist- und Soll-Prozesse nicht klar genug hervorgingen. Beispiele für zunächst übersehende Anforderungen sind die Terminverwaltung eines Prüfers und das Erstellen von Vorlesungsdaten. Weiterhin ist zu bemerken, dass es nicht immer gelungen ist, den Detailgrad der Anforderungen passend zu wählen. So sind einige Anforderungen so allgemein gehalten, dass sie keine zusätzlichen Informationen über das zu implementierende System enthalten, während andere Anforderungen zu sehr ins Detail gehen.
Trotz dieser Kritikpunkte bleibt festzuhalten, dass es der PG gelungen ist, die wesentlichen Anforderungen an das zu entwickelnde System zu erfassen und in übersichtlicher Form festzuhalten. Insbesondere hat die vorgeschaltete Dokumentationsphase, bei der eine Vielzahl von Interviews mit den an den Prozessen beteiligten Personen stattgefunden hat, maßgeblich zum Prozessverständnis und damit auch zur Anforderungsanalyse beigetragen.

10.1.3 Grob- und Feinentwurf des Systems

Da der Grobentwurf des Systems im Wesentlichen von der J2EE Architektur vorgegeben war, bestand die einzige Aufgabe der PG darin, entsprechende Komponenten wie Container und Datenbank auszuwählen und miteinander zu verzahnen. Diese Aufgabe wurde zufrieden stellend gelöst.
Beim Übergang vom Grobentwurf zum Feinentwurf wurde verstärkt Gebrauch von Design Patterns gemacht. Die ausgesuchten Entwurfsmuster wie z.B. Session Facade, Business Service Locator und Value Object Assembler trugen maßgeblich zu einem soliden Feinentwurf des Systems bei. Auf dessen Basis war eine Implementierung möglich. Es erwiesen sich jedoch nicht alle ausgesuchten Entwurfsmuster und die damit verbundenen Design-Entscheidungen als durchführbar. Dieser Umstand und die Tatsache, dass die verwendete J2EE Technologie den Teilnehmern der PG nicht oder nur unzureichend vertraut war, führte dazu, dass Änderungen am Feinentwurf noch in späteren Phasen der Entwicklung notwendig waren. Betrachtet man jedoch andere Softwareprojekte, so scheinen relativ späte Änderungen am Entwurf durchaus des Öfteren notwendig zu sein. Deshalb ist es nicht weiter verwunderlich, dass diese Notwendigkeit sich innerhalb der PG ergeben hat.
Alles in allem war der in dieser Phase der PG entstandene Entwurf ausreichend geeignet, um auf dessen Basis ein lauffähiges System zu implementieren, das einen Großteil der zuvor festgelegten Anforderungen erfüllt.

10.1.4 Implementierung eines Systems

Vergleicht man das entwickelte System mit den zuvor festgelegten Anforderungen, dann ist festzustellen, dass nicht alle Anforderungen erfüllt werden konnten. Die Gründe hierfür liegen unter anderem darin, dass die Teilnehmer der PG mit den genutzten Techniken wie z.B. der J2EE Architektur oder Struts von vornherein nicht ausreichend vertraut waren. Dies führte dazu, dass Anforderungen ein wenig „blauäugig“ definiert wurden, die sich später gar nicht oder nur mit sehr großem Aufwand realisieren ließen. Ein Teil der Anforderungen konnte durch die mangelnde Kooperationsbereitschaft seitens des Herstellers des Hochschulinformationssystems [HIS] nicht erfüllt werden.
Ebenso ist festzuhalten, dass die entstandene Software nicht in dem Maße getestet worden ist, so dass sie einen sofortigen Einsatz ermöglicht. Hierzu wären eine ausführliche Testphase sowie weitere Feinarbeit notwendig. Es ist der PG jedoch gelungen, einen Prototyp zu entwickeln, der einen Großteil der definierten Anforderungen abdeckt und einige Möglichkeiten bietet, die über die zuvor definierten Anforderungen hinausgehen. Betrachtet man weiterhin, dass mit J2EE eine relativ neue Architektur eingesetzt worden ist, so kann man mit dem entstandenen Ergebnis sehr zufrieden sein.

10.2 Extras und außerplanmäßige Zusatzleistungen

10.2.1 Freies System

Clevery ist in seinem jetzigen Stadium fast komplett mit frei verfügbaren Tools zu generieren und weiterzuentwickeln. Dies war zwar keine direkte Zielsetzung, aber als Seiteneffekt sehr willkommen.

10.2.2 Der Artikel

Zu Beginn der PG im WS 01/02 wurde beschlossen, Informationen über das Projekt nach außen zu kommunizieren. Dabei wurde Inhalte und Medien weitgehend offen gelassen.
Zu Beginn des SS 02 zeichneten sich die Konturen des Systems soweit ab, dass genügend Material für einen Bericht zur Verfügung stand. Als ideales Forum für die Bekanntmachung des Clevery-Projekts bot sich das Magazin „UniReport“ an. Der UniReport richtet sich in erster Linie an universitätsinterne Einrichtungen wie Lehrstühle sowie Mitarbeiter und angegliederte Bereiche. Darüber hinaus wird das Magazin an die Industrie verteilt. Interessant am UniReport ist nicht nur die Adressatenstruktur, sondern auch die hochwertige Aufmachung. Durch sein Selbstverständnis als Medium für die Darstellung hochschulinterner Forschung an der Universität Dortmund bot es für eine Veröffentlichung der Ergebnisse und Perspektiven der Projektgruppe ein ideales Forum. Die kostenlose Verfügbarkeit und eine Auflage von 5.000 Exemplaren sorgen im Bereich der Universität für eine hohe und zielgruppengerechte Verbreitung.
Die Ausrichtung der Inhalte sollte vor allem den Forschungsbetrieb an der Universität illustrieren. Dabei blieb es den Autoren offen, inwiefern sie auf Forschungsdetails eingehen. Aus diesem Grund wurde der Artikel entsprechend optisch gefällig gestaltet. Mit freundlicher Unterstützung der Presseabteilungen von „vodafone“ und der Organisation „JBOSS Group“, die uns Pressematerial in Form von Bildern zur Verfügung stellten, und dem Photograph der Abteilung Öffentlichkeitsarbeit an der Universität Dortmund, der das nightshift-Teams professionell ablichtete, konnte der Artikel auffällig aufbereitet werden.
Statt einer Schilderung der Inhalte wird im Folgenden der Artikel in seiner Erscheinungsform gezeigt. Für das letztendliche Layout ist die Redaktion des UniReports verantwortlich. Die textlichen Inhalte sowie die Auswahl und Anordnungen der einzelnen Grafiken lagen in den Händen des nightshift-Teams. Veröffentlicht wurde der Artikel in der Ausgabe 34/2002, Seite 58/59, ISSN 0179-7182. Reaktionen von außen auf den Artikel blieben bislang leider aus.

MAGAZIN

Just be Clevery!

Infratechnologie für den Universitätseinsatz

Oliver Effner und Prof. Dr. Volker Gruhn, Lehrstuhl für Informatik 10

Ein Hochschulstudium enthält für die Studierenden viele Hürden. Neben den eigentlichen Inhalten erschweren Bürokratie und Organisation den zügigen Studienablauf. Mit „Clevery" haben Informatiker der Universität Dortmund jetzt eine Software entwickelt, die das Alltagsgeschäft von Studierenden und Universitätsmitarbeitern durch den Einsatz moderner Kommunikation automatisiert und optimiert.

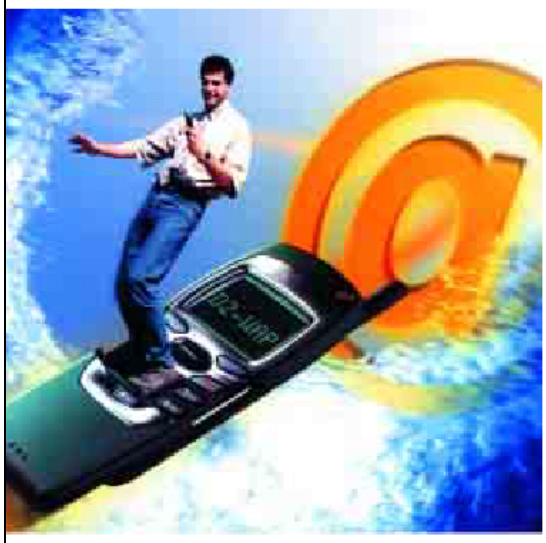


Abbildung 1: Mit WAP und Internet schneller durch Studium (Foto: Vodafone)

In der letzten Zeit wird vor allem von Seiten der Industrie immer wieder gefordert, die Informatikausbildung an den Hochschulen zu beschleunigen, um die Nachfrage an qualifiziertem IT-Personal decken zu können. In der Tat liegt die durchschnittliche Studiendauer im Fachbereich Informatik an der Universität Dortmund bei ca. 14,8 Semestern. Nur 30 Prozent der Erstsemestler schaffen es, in den veranschlagten acht Semestern ihr Vordiplom zu erhalten. Wird zusätzlich noch in Betracht gezogen, dass die Zahl der Neueinschreibungen in der letzten Zeit bei gleich bleibenden Personal- und Raumressourcen enorm zugenommen hat, dann sind Staus und Engpässe unvermeidlich. Dies wiederum wirkt sich auch nachteilig auf die Studiendauer aus.

Als schnelle Reaktion auf diese Umstände wurde am Fachbereich Informatik das WIS-Projekt entwickelt. Dieses Sofortprogramm, gefördert durch die Zinsersparnisse aus den Versteigerungserlösen der UMTS-Mobilfunklizenzen, umfasst ein breit angelegtes Maßnahmenbündel, um das Informatikstudium an den deutschen Hochschulen weiterzuentwickeln. Ein Strang dieses Bündels ist das Projekt „Clevery". Initiiert wurde es unter der Leitung von Prof. Dr. Volker Gruhn. Der Name „Clevery" entstand aus der Kombination der Worte „close" und „everywhere" und unterstreicht die für das Projekt zentrale Dezentralität. Gemeint ist die Auflösung der bisherigen Ortsabhängigkeit: Organisatorisches muss nicht mehr vor Ort an der Universität erledigt, sondern kann bequem von zu Hause abgewickelt werden. Der Fokus des Projektes liegt aber vor allem auf einer Verbesserung der Studienorganisation durch eine Optimierung der Abläufe am Fachbereich Informatik. Die Realisierung erfolgte in zwei Stufen.

Die erste Stufe begann mit der Modellierung realer Abläufe, also der Ist-Zustände sowie ihrer Analyse. Nach der Analysephase zeichnete sich ab, dass es hauptsächlich zwei Komplexe gibt, bei denen eine Optimierung zu deutlichen Verbesserungen führen würde. Der erste Komplex betrifft die Organisation von Prüfungen, der zweite die Organisation von Übungsgruppen. Zu dieser Erkenntnis führten eigene Erfahrungen sowie Interviews und Umfragen, die hauptsächlich mit den Angestellten der einzelnen Lehrstühle durchgeführt wurden. Ein klassisches Beispiel für die Vorteile einer verbesserten Organisation ist die Anmeldung zu den Übungsgruppen der Erstsemestervorlesungen, bei denen teilweise mehrere hundert Studierende gleichzeitig versuchen, sich in die aushängenden Listen einzutragen. Zu diesen Drängeleien kommt hinzu, dass viele Studierende eventuellen Terminkollisionen dadurch begegnen, dass sie sich mehrfach in die Listen eintragen. Ein Verfahren, dass den Zeitaufwand für die Bearbeitung und Auswertung der Listen bei den einzelnen Lehrstühlen erheblich erhöht. Zudem ergeben sich Probleme bei der fairen Zuteilung. Diese Ineffizienzen führen dazu, dass Professoren und Mitarbeiter durch organisatorische Aufgaben belastet werden und weniger Kapazität für fachliche und inhaltliche Aufgaben in Forschung und Lehre übrig bleibt.

Clevery soll genau an dieser Stelle ansetzen. Bequem soll sich der Studierende dann per Uni-Terminal, privatem PC oder auch per WAP-Handy zu einer Veranstaltung an- und abmelden können. Anschließend unterstützt Clevery nach Ablauf der Anmeldefrist dann die Mitarbeiter der Universität bei der Raumzuteilung und der Einteilung der Übungsgruppenleiter. Dabei steht eine faire Verteilung im Vordergrund und nicht das bisher praktizierte First-Come-First-Serve-Prinzip. Weiterhin soll der Ablauf der Veranstaltungen optimiert werden. So ist geplant, die Kommunikation zwischen den Veranstaltern und den Studierenden zu verbessern. Findet eine Veranstaltung kurzfristig nicht statt oder gibt es Raumverschiebungen, braucht dies nur eingegeben zu werden und die Benachrichtigung der betroffenen Teilnehmer wird dann vom System übernommen. Dies ist nur ein Beispiel für das Verbesserungspotenzial. Auch die Übersicht über das eigene Studium an sich ist ein wichtiger Aspekt, den Clevery umsetzen soll, indem es Informationen zu den eigenen laufenden und abgeschlossenen Veranstaltungen aufbereitet. Weiter-

Abbildung 10-1: UniReport erster Teil

hin gibt es auch Visionen, die sich direkt durch die Möglichkeiten einer solchen Technologie ergeben. Vorstellbar ist auch, Übungszettel zu bearbeiten und weitestgehend automatisch zu korrigieren oder Fragen innerhalb einer Übungsgruppe zur Diskussion zu stellen.

In der zweiten Stufe wird ein intranetbasiertes System entwickelt, das die Angehörigen des Fachbereichs – Studierende wie Lehrende – bei der Durchführung der optimierten Prozesse unterstützt. Das entwickelte System soll per Internet und insbesondere auch mit mobilen Endgeräten, wie WAP-fähigen Mobilfunktelefonen, orts- und zeitunabhängig zu bedienen sein. Diese Zielsetzung setzt eine leistungsfähige softwaretechnische Infrastruktur voraus. Zum einen reicht die Funktionalität eines herkömmlichen Web-Servers nicht aus, zum anderen muss eine einfache und vor allem schnelle Implementierung des Systems möglich sein. Dabei darf auch die Kostenfrage nicht vernachlässigt werden. Als Lösung bietet sich eine komponentenbasierte Applikationsumgebung an. Bei einer solchen Umgebung steht ein Applikationsserver im Mittelpunkt, auf dem das komplette System läuft. Das hat den Vorteil, dass die Anforderungen an die Endgeräte minimal sind, so dass für viele Anwendungsfälle ein im Vergleich zu PCs extrem leistungsschwaches WAP-Handy ausreicht. Die eigentliche Verarbeitung der Daten geschieht gänzlich auf dem Applikationsserver. Zudem bieten Applikationsserver die gesamte Infrastruktur, wie beispielsweise die Anbindungen von Datenbanken oder die Implementierung von Sicherheitsmechanismen, so dass bei der Implementierung auf fertige Strukturen zurückgegriffen werden kann und die Entwickler sich auf die eigentlichen Aufgaben konzentrieren können. Um die Projektanforderungen umzusetzen, kann die Software weiter in verschiedene, autonome Teile zerlegt und die Gestaltung der Benutzerschnittstelle separat entworfen werden. Diese Konzeptionen werden mit Hilfe der Java Enterprise Technologie von Sun umgesetzt. Ein weiterer Vorteil

Abbildung 2:
Das Clevery Team

dieser Technologie ist die freie Verfügbarkeit. Das gilt auch für die Applikationsserver, die sich zwar in vielen Belangen nicht mit kommerziellen Produkten messen können, aber für die Zielsetzung ausreichen. Clevery basiert auf dem freien Applikationsserver Jboss, der kostenfrei unter http://www.jboss.org zu beziehen ist. Ein Applikationsserver erfüllt natürlich nicht alle Anforderungen des Projektes, sondern muss durch mehrere weitere Softwarepakete ergänzt werden. Clevery benutzt aus diesem Grund zusätzlich die in der Abbildung 3 dargestellten Komponenten der ebenfalls frei verfügbaren Programme Tomcat und Postgres. Ganz auf kommerzielle Produkte konnte dennoch nicht verzichtet werden. Für die Implementierung wird auf eine professionelle Entwicklungsumgebung zurückgegriffen, und auch die Modellierung und Optimierung in der ersten Phase wurden mit kommerzieller Software durchgeführt. Da einige Technologien in dieser Version noch nicht unterstützt werden, wurde der Applikationsserver in der derzeitig aktuellen Version ebenfalls durch verschiedene kommerzielle Komponenten ergänzt. Dies wird sich in Zukunft mit der neuen JBoss 3.0 Version ändern.

Die Industrie ist sehr an dem Projekt interessiert. Die softwaretechnische Umsetzung (insbesondere Studien im Bereich des Usability Engineering) werden von der Siemens AG unterstützt. Die Siemens AG wird den Teilnehmern der Projektgruppe zur Entwicklung der entsprechenden Anwendung mobile Endgeräte zur Verfügung stellen.

Infos:
http://www.pg-nightshift.de

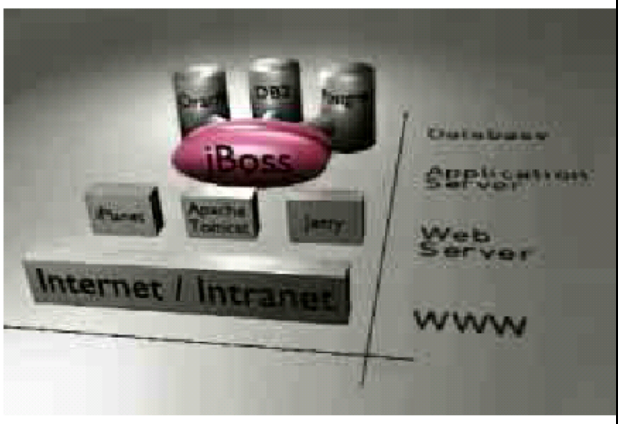


Abbildung 3:
Das Herz von Clevery
(Foto: JBoss Group LLC)

Abbildung 10-2: UniReport zweiter Teil

10.2.3 Das Campusfest

Eine erste Präsentation des Systems wurde auf dem Campusfest der Universität im Sommer 2002 gezeigt. Für die Demonstration des Systems wurde ein Stand aufgestellt, auf dem im Schichtbetrieb zwei Teams aus PG-Mitgliedern die Software präsentierten.
Die Vorbereitungen für diese Veranstaltung begannen schon am Ende des Wintersemesters 01/02 und wurden im Rahmen des Fachbereichs Informatik koordiniert. Ein Gremium der einzelnen Aussteller (Lehrstühle und Projektgruppen) planten in dieser Phase grob die Aufstellung der zu präsentierenden Projekte. Zusätzlich wurde über die Bekanntmachung der jeweiligen Projekte und über mögliche Promotion-Aktionen nachgedacht.
Der übergeordnete Rahmen für das Campusfest war eine komplette Darstellung des Fachbereichs Informatik mit zusätzlicher Studienberatung. Die Adressaten dieser Veranstaltung waren weniger das Fachpublikum als die studieninteressierten Schulabgänger dieses Jahrgangs.

10.2.3.1 Der Stand

Die Stände waren traditionsgemäß im unteren Teil des Audimax-Foyers aufgestellt. Die Zuweisung der Standplätze erfolgte per Zufallsprinzip. Zwar gab es noch Spielraum für besondere Wünsche, aber mangels Erfahrung waren alle Aussteller mit der Zuweisung einverstanden. Der PG-nightshift wurde der Stellplatz direkt am unteren Ende der Treppe zugewiesen, was sich als erheblicher Nachteil herausstellte. Das Problem lag darin, dass die direkt am Geländer befestigten Eyecatcher (Poster, Beamer) nicht von Leuten, die die Treppe benutzten, wahrgenommen werden konnten. Des Weiteren versperrten zwei Stützpfeiler die Sicht der Interessenten, die das Audimax über den unteren Eingang am Parkplatz betrat.
Für die eigentliche Vorführung wurden ein Rechner mit Monitor, ein Beamer und eine zusätzliche Leinwand besorgt. Von den Organisatoren wurden für jeden Stand eine Stellwand, ein Tisch und zwei Stühle bereitgestellt.
Wie eingangs erwähnt bestand die Zielgruppe vorwiegend aus Studieninteressenten. Das war das Kriterium, weswegen ein sehr eigenwilliger Weg bei der Präsentation eingeschlagen wurde. Statt heller, überwiegend technischer Präsentation von Diagrammen und Tabellen, wurde eine stärker zielgruppengerechte Aufmachung gewählt. Die Kernaussage des Standes war ein Überblick über die Kommunikationsstruktur und den rollenbasierten Ansatz der Software. Dafür wurden Trendelemente aufgegriffen, die dem Interesse der Zielgruppe entsprachen: Handys und Star Wars. Die dahinter stehende Idee war, das Publikum neugierig zu machen und an den Stand zu locken. Deshalb wurde auf diese „Gegenwartsikonen“ zurückgegriffen: Star Wars Episode II war erst kurz vorher in den Kinos angelaufen und Handys sind für diese Zielgruppe sehr interessant.

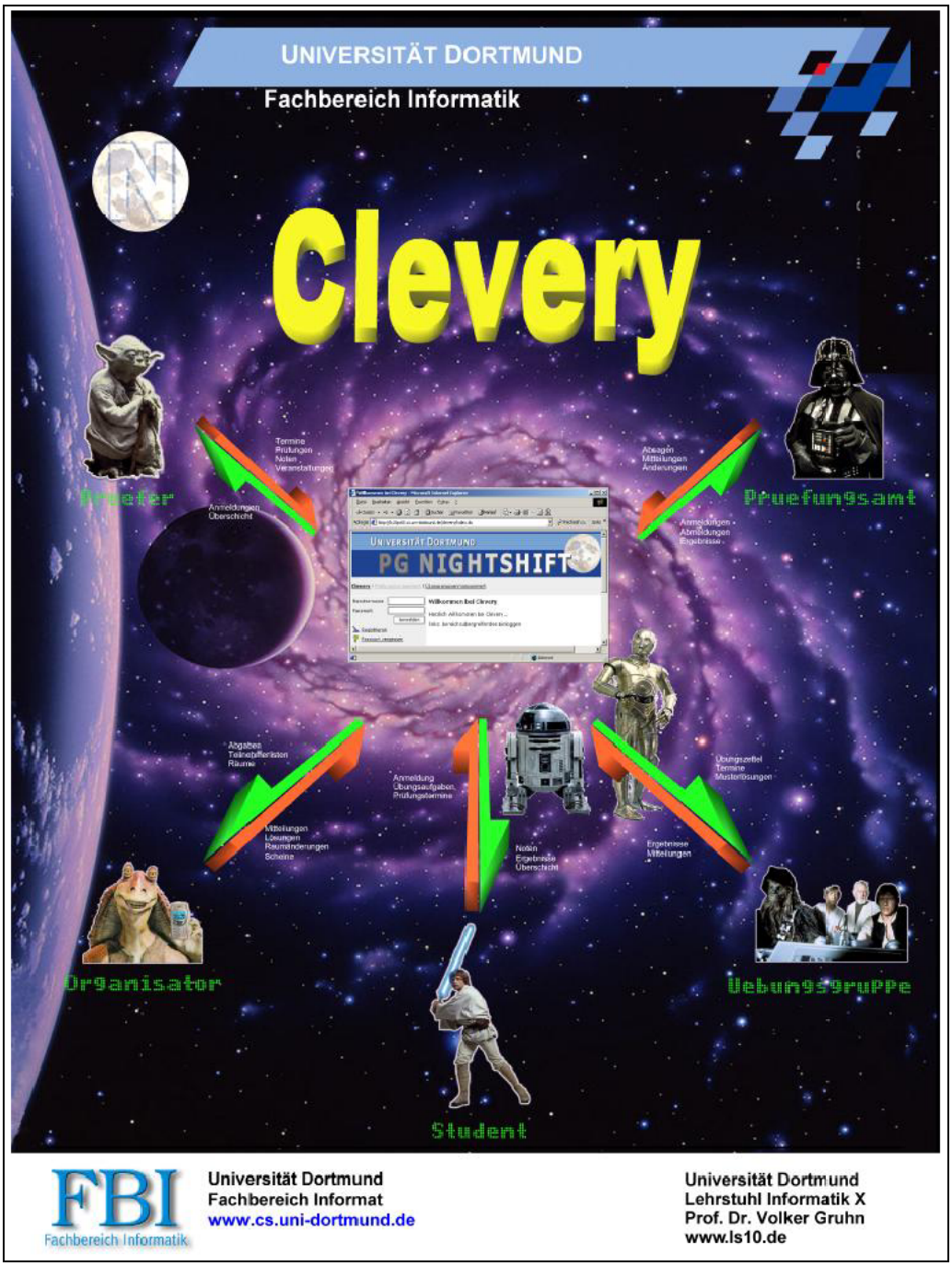


Abbildung 10-3: Plakat für das Campusfest

Als echtes Novum waren in die Power Point Präsentation, die an dem Stand gezeigt wurden, Sounds und Klangeffekte eingebaut. Als Songs wurde der aktuelle Hit „Bohemian like you“ von den Dandy Warhols, besser bekannt als „vodafone-Song“ aus der Werbung, und das ebenfalls sehr bekannte Thema von Star Wars I (Episode IV) ausgewählt. Leider war es nicht möglich während des Campusfests das Soundvolumen so zu wählen, dass die soundtechnische Untermalung für die gewünschte Wirkung sorgte.
Als besonders Bonbon konnten alle Interessenten die Funktionalität von Clevery per Handy ausprobieren. Wer nicht sein eigenes Handy benutzen wollte, konnte die Handys am Stand nutzen, die uns für diese Veranstaltung von der Firma Siemens kostenlos zur Verfügung gestellt wurden. Es handelte sich dabei um den gängigen Gerätetyp ME 45, auf dem das System auch vorher getestet wurde.

Den letzten Schliff gab eine großzügige Spende der Firma Apollinaris, die uns unentgeltlich mit Getränken unterstützte, die gratis an alle Besucher und an andere Aussteller ausgegeben wurden.

10.2.3.2 Bewertung

Die erhoffte Aufmerksamkeit konnten wir leider nicht auf uns lenken. Viele Faktoren erschwerten dies. Jedoch wäre es falsch, allein die Gründe dafür bei der Projektgruppe und dem Standkonzept zu suchen. Andere Projekte, die ebenfalls in diesem Bereich ausgestellt wurden, hatten mit der gleichen Problematik zu kämpfen. Organisatorisch ließe sich der Auftritt des Fachbereichs Informatik sicher besser vermarkten. Das Hauptproblem allerdings war das fehlende Basisinteresse beim Publikum. Hätte es einen publikumswirksamen Event gegeben, der als Leitmotiv den Informatikbereich aktiviert hätte, in dessen Wirkungsbereich die einzelnen Stände sich hätten präsentieren können, so wären bestimmt weit weniger Gratisflaschen übrig geblieben.
Dennoch war die Entscheidung sich auf dem Campusfest zu engagieren sehr gut. Die Präsentation als Ziel sorgte für einen gewaltigen Schub bei der Entwicklung des Systems. Viele Anforderungen an clevery, die zu Beginn der PG festgelegt wurden, sind vor allem in der Vorbereitung auf das Campusfest implementiert worden, auf dem sie dann auch erstmalig der Öffentlichkeit vorgestellt wurden. Die Intensität der Entwicklungsarbeit und der Einsatz des gesamten Teams waren in dieser Zeit am höchsten und brachten die Ziele ein großes Stück näher.

11 Fazit

Nach jeder Projektarbeit ist es sinnvoll und notwendig den gesamten Prozess kritisch zu betrachten und zu bewerten. Dies soll in diesem Kapitel geschehen.
Um dies zu tun, werden verschiedene Gesichtspunkte, wie zum Beispiel organisatorische Probleme, Gruppendynamik und externe Unterstützung, untersucht. Dabei soll aber nicht nur negativ kritisiert, sondern auch positive Punkte herausgearbeitet werden.
Dazu gibt es zunächst eine allgemeine Kritik der Projektgruppe. Danach werden bestimmte Probleme in Ablauf, Organisation und Zusammenarbeit betrachtet. Zu guter letzt kommen positive Aspekte und Erfolge der Projektgruppe zur Diskussion.

11.1 Allgemeine Kritik

Nach Betrachtung des ganzen Prozesses ergaben sich folgende allgemeine Problemstellungen:

- **Unzureichende Ausstattung mit Räumen und Hardware**
  Leider war die Ausstattung der PG recht dürftig. Zunächst dauerte es fast 4 Monate, bis den Projektteilnehmern Rechner zur Verfügung gestellt wurden. In der Zwischenzeit musste auf private Ressourcen der Teilnehmer zurückgegriffen werden. Allerdings bekam die Gruppe nach diesen 4 Monaten nur 1 Server und 4 Workstations, was sich bei 11 Teilnehmern als viel zu wenig herausstellte. Zudem war der PG-Raum für gleichzeitiges Arbeiten aller Beteiligten viel zu klein. Dieses Problem wurde gelöst, indem ein Belegungsplan erstellt wurde und einige Teilnehmer zu Hause arbeiteten. Leider kam es durch diese räumliche und zeitliche Trennung zu Kommunikationsproblemen. Daraus resultierten unnötige Verzögerungen und Missverständnisse. Diese mussten in den regelmäßigen Treffen korrigiert werden.
- **Unflexible Fixierung auf harte Bedingungen bzw. den Prüfungsteil**
  Sehr früh in der Projektarbeit stellte sich heraus, dass eine sinnvolle Umsetzung der Prüfungsanmeldung in Zusammenarbeit mit der HIS nicht möglich sein würde. Daraufhin wurde von einigen Teilnehmern der Vorschlag gemacht, den Prüfungsteil komplett fallen zu lassen und dafür den Übungsgruppenteil zu erweitern. Dies wurde aber von den Veranstaltern der Projektgruppe abgelehnt. Daher wurde eine Lösung für den Prüfungsteil entwickelt, die im eigentlichen Sinne überflüssig ist. Dies hatte natürlich negative Auswirkungen auf die Motivation der Teilnehmer.

11.2 Probleme

11.2.1 PG (Zusammenarbeit, Organisation)

Folgende Probleme im Bereich Zusammenarbeit und Organisation haben sich im Verlauf der Projektgruppe ergeben:

- **Ungleichmäßige Auslastung der Teilnehmer**
  Im Verlauf der Projektarbeit stellte sich schnell heraus, dass einige Teilnehmer mehr Aufgaben übernahmen als andere. Dies ist gerade am Anfang eines Projekts, aufgrund der unterschiedlichen Vorkenntnisse der Teilnehmer, ganz natürlich. Allerdings sollte sich das nach einiger Zeit relativieren. Dies war leider bei einigen Teilnehmern nicht der Fall. Worauf das genau zurückzuführen ist, lässt sich an dieser Stelle schlecht beantworten. Hier kann man sich allerdings auch nicht damit herausreden, dass man bestimmte Dinge nicht kann oder kennt. Schließlich ist die Projektarbeit gerade dazu da, neue Dinge zu lernen. Glücklicherweise eskalierte diese Situation jedoch nicht und die meisten Teilnehmer waren auch voll bei der Sache. Nichtsdestotrotz hinterlässt ein solches Verhalten bei den stärker involvierten Teilnehmern einen bitteren Nachgeschmack.
- **Sitzungen wurden schlecht durch die Teilnehmer moderiert**

In den regelmäßigen Sitzungen wurde reihum ein Protokollführer bestimmt, der ein Protokoll anlegte und in der nächsten Sitzung die Moderation der Diskussion übernahm. Leider kam es trotzdem zu zeitraubenden und oftmals auch unnützen Diskussionen, wobei der Moderator es nicht verstand diese zu beenden oder in eine sinnvollere Richtung zu dirigieren. Auch dies gehört zu den Lernzielen einer Projektgruppe.

- **Probleme beim gleichzeitigen Bearbeiten von Dateien**
  Bei einem Projekt an dem mehrer Personen mitarbeiten, kommt es vor, dass Dateien gleichzeitig von mehreren Personen bearbeitet werden müssen. Dafür gibt es mehrere Lösungsansätze. Die Projektgruppe entschied sich für den Einsatz von CVS. Dies brachte allerdings auch einige Probleme mit sich. Da die meisten Dokumente mit Word verfasst wurden, eignete sich das CVS-System nicht zur parallelen Bearbeitung. Auch die umständliche Bedienung und einige Bugs im CVS-System führten zu Fehlern. Daraus ergibt sich, dass man genau planen sollte, wie man seine Dateien und deren Bearbeitung organisiert.
- **Unklare Aufgabenverteilung**
  Während des Ablaufs der Projektgruppe war es einigen Teilnehmern nicht klar an welche Aufgabe sie sich heranwagen durften. Oftmals wurde so lange gezögert, bis eine Aufgabenstellung unvermeidlich war. Nur wenige der Teilnehmer waren bereit die Initiative zu ergreifen und einen Vorschlag zu machen, um die Arbeit voranzutreiben. Solch eine „passive Mitarbeit“ kostet viel Zeit.
- **Terminplan**
  Wir hatten einen sehr präzise ausgearbeiteten Terminplan, um zu garantieren, dass wir stetig Fortschritte machen. Diese gute Idee wurde aber von den meisten von uns ignoriert. Ein solches Projekt erfordert gegenseitige Absprache, aber das Privatleben (Arbeit und familiäre Angelegenheiten) hat in manchen Hinsichten auch Vorrang. Eine genaure Einteilung der Zeit in Entwicklungsphasen ist also sehr schwierig zu machen, was wir ziemlich bald feststellen mussten.

11.2.2 Ablauf

Folgende Probleme im Ablauf der Projektarbeit haben sich ergeben:

- **HIS lehnt Zusammenarbeit ab**
  Ziemlich früh im Projektablauf stellte sich heraus, dass die HIS die Zusammenarbeit mit der Projektgruppe ablehnt. Dadurch zeichnete sich schon ab, dass das zu entwickelnde System nie zum Einsatz kommen würde. Dies wirkte sich extrem negativ auf die Motivation der Teilnehmer aus.
- **Die Sache mit Viren**
  Leider wurden alle Workstations zu einem denkbar schlechten Zeitpunkt im Projektablauf von einem Virus befallen. Zwar wurden schnell Maßnahmen zur Bekämpfung ergriffen, aber trotzdem kam es zu einigen Schäden und zu Zeitverlusten.
- **Strom- und Netzausfälle**
  Zu weiteren Zeitverlusten führten die häufigen Strom- und Netzausfälle an der Universität. Allerdings hatte die Projektgruppe hierauf keinen Einfluss.
- **Hardwareausfälle**
  Wie schon erwähnt, war die Ausstattung der Projektgruppe recht mager. Zusätzlich wurde die Situation noch durch mehrere Hardwareausfälle verschärft. Allerdings konnte der Hardware-Verantwortliche zusammen mit der Universität recht schnell Abhilfe schaffen.
- **Kommunikation zwischen Geschäftlogik- und GUI- Bereich problematisch**
  Die Realisierung des Systems wurde in zwei Bereiche aufgeteilt: In den Bereich Geschäftslogik und in den GUI-Bereich. Die Aufteilung war auf jeden Fall sinnvoll, allerdings erforderte dieses Vorgehen auch einen Mehraufwand an Kommunikation. Hier kam es allerdings zu Problemen. Diese wurden durch das Raum- und

Ressourcen-Problem noch verschärft. Zunächst wurde versucht die Kommunikation über E-Mail abzuwickeln, was sich als zu langsam erwies. Danach wurde das Tool Wiki eingesetzt, was etwas Abhilfe brachte. Aber erst, als Zweierteams aus jeweils einem Teilnehmer des GL-Bereichs und der GUI gebildet wurden, ging die Arbeit richtig voran.

11.3 Erfolge / Positives

Nach den ganzen Problemen, die sich im Laufe der Projektgruppe ergeben haben, soll nun aber auch auf die positiven Entwicklungen verwiesen werden.

- **Viel Neues gelernt**
  Als ein positives Ergebnis ist wohl die Tatsache zu nennen, dass jeder Teilnehmer eine Menge Neues in vielen Gebieten, wie zum Beispiel Technik, Teamwork und Organisation, gelernt haben sollte. Wenn es Teilnehmer gibt, bei denen dies nicht der Fall ist, haben diese etwas falsch gemacht.
- **Einblick in J2EE Welt**
  Für viele Teilnehmer der Projektgruppe war ein wichtiges Ziel, Einblick in die J2EE Technologie zu bekommen. Dazu gab es in ausreichendem Maße Gelegenheit.
- **Einarbeitung in weitere Technologien**
  Den Projektgruppenteilnehmern war es möglich sich in weitere Technologien einzuarbeiten. Struts, JAAS-, JavaMail- und Reflection-API sind nur einige davon. Dieses Wissen sollte sich im weiteren Berufsleben als hilfreich erweisen.
- **Erfahrungen mit der Projektarbeit in einem größeren Team**
  Wichtig sind auch der Erfahrungen, die sich aus der Teamarbeit ergeben haben. Gerade die Probleme, die entstehen, wenn man sich bei der Arbeit mit anderen Personen arrangieren muss, sind eine wichtige Erfahrung. Auch die gemeinsame Problemfindung und -lösung ist ein wichtiger Lernprozess, der nicht erst im späteren Berufsleben in Angriff genommen werden sollte.
- **Hilfsbereitschaft der anderen Lehrstühle und Einrichtungen**
  Gute Erfahrungen haben wir ebenfalls während der Interviews mit dem Lehrstuhl V gemacht. Einige Auskünfte über das Templus Projekt haben uns weitergebracht. Ebenso war unser Rechenzentrum bei der Lösung unserer Probleme mit dem Authentifizierungsteil sehr hilfsbereit. Hiermit möchten wir uns auch bei ihnen bedanken.
- **Einen Dank auch an die Firmen, die uns unterstützt haben**
  Dazu zählt die Firma Adesso, deren Programm „LeuSmart“ nützlich beim Erstellen der informativen Diagramme war. Nach anfänglichen Kommunikationsschwierigkeiten (und Problemen beim Brennen eines Testexemplars) haben wir schließlich ihr Produkt zum laufen bekommen.
  Noch ein großes Dankeschön an die Firmen, die uns mit Getränken auf dem Campusfest versorgt haben. Es hat uns mehr interessierte Zuschauer gebracht. Es ist wirklich erstaunlich wie weit man mit Freundlichkeit kommt und was man dabei alles erreichen kann.

11.4 Ausblick

In der Entwicklungsphase von Clevery wurden viele Gespräche und Interviews geführt, die die Zukunftsaussichten des Projektes bereits früh aufgezeigt haben. Bei Recherchen zum Prüfungsmanagementprozess stellte sich heraus, dass die Hochschulinformationssysteme (HIS) GmbH, die in Kooperation mit Bund und Ländern viele Hochschulen in ganz Deutschland mit ihrer Studenten- und Prüfungsverwaltungssoftware beliefert, bereits einen Softwareprototyp für Prüfungsprozesse anbietet und an der Universität Dortmund im Fachbereich Elektrotechnik im Testeinsatz betreibt. Seitens der HIS GmbH wurde, nachteilig für die PG-nightshift , keine Kooperation angestrebt. Obwohl theoretisch durchaus eine softwaretechnische Anbindung von Clevery an vorhandene Systeme in Betracht gezogen werden kann, werden die realisierten Prüfungsmanagementprozesse mit hoher Wahrscheinlichkeit nicht in den laufenden Betrieb integriert.

Aus den Übungsgruppenmanagement-Funktionen von Clevery hingegen könnte der Fachbereich Informatik Nutzen ziehen, wenn ein Team von Entwicklern zur Pflege und Weiterentwicklung gebildet würde. Den Clevery-Entwicklern sind während der einjährigen Entwicklungszeit diverse Ideen für Erweiterungen gekommen, die den Verwaltungsaufwand vieler Prozesse im Fachbereich verringern kann. Ein Raum-/Zeitmanagementsystem mit automatisiertem Raumzuteilungsverfahren beispielsweise könnte den bisherigen fehleranfälligen und manuell durchgeführten Prozess ersetzen. Die Integration eines zentralen Informationssystems auf Content Management System-Basis, das z.B. Veranstaltern, Organisatoren, aber auch Studenten die Möglichkeit gibt, fachbereichsspezifische Inhalte wie Veranstaltungsdaten, Projektgruppen- und Seminarvortreffen zu erstellen und zu veröffentlichen, würde unterstützend dazu beitragen, die an der Universität Dortmund oft langwierige Informationsbeschaffung aller beteiligten Rollen zu vereinheitlichen. Als weitere Ausbaustufe könnte eine offene Kommunikationsplattform geschaffen werden, die zusätzlich zum reinen Informationsangebot beispielsweise die Möglichkeit für Rückfragen und Diskussionen, Speicherung beliebiger persönlicher Daten wie E-Mails, Termine und Notizen sowie Speicherung gemeinsamer Daten für geschlossene Nutzerkreise bietet.

Mit Clevery ist der softwaretechnische Grundstein für weitere Projekte zur Optimierung der Geschäftsprozesse am Fachbereich Informatik gelegt, die Umsetzung sei Mitarbeitern der Lehrstühle oder folgenden Projektgruppen überlassen.

Literaturverzeichnis

[ant] Apache Software Foundation 1999-2002. Apache Ant 1.5.1 Manual. http://jakarta.apache.org/ant/manual/index.html. Version 1.5.1/ Juni 2002

[apache] Apache Software Foundation.1999-2002. Apache HTTP Server Projeckt. http://httpd.apache.org/docs-project . Januar 2002

[beans] Sun Microsystems, Inc. 1995-2002. THE ONLY COMPONENT ARCHITECTURE FOR JAVA™ TECHNOLOGY. http://java.sun.com/products/javabeans. Januar 2002

[Deck-it] Bezugsquelle: http://www.pyweb.com/tools/ (März 2002)

[DepTool] „JBoss Deployer Plugin for Together ControlCenter“ Version 1.0.6 von M. J. Swainston-Rainford (für $60 beziehbar) www.flashline.com

[DP] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides Design Patterns. Elements of Reusable Object-Oriented Software. 1994. Boston. Addison-Wesley Verlag, ISBN: 0201633612

[GruSch02] Volker Gruhn, Manfred Schneider. 2002. EJB 2.0 Anwendungen. München. Addison-Wesley Verlag, 2002. ISBN: 3827319773

[HIS] Hochschul-Informations-System GmbH. 2002.Softwarehaus der Hochschulverwaltungen . http://www.his.de/Abt1. November 2001

[jak] Apache Software Foundation 1999-2002. The Jakarta Project. http://jakarta.apache.org/tomcat/index.html. März 2002

[JEE2-v1.3] Sun Microsystems, Inc. / Bill Shannon. August 22, 2001. Java™ 2 Platform, Enterprise Edition (J2EE™) Specification, v 1.3. http://java.sun.com/j2ee/download.html. Dezember 2001

[jsp] Sun Microsystems, Inc. 1995-2002. JavaServer Pages – Dynamically Generated Web Content. http://java.sun.com/products/jsp. August 2002

[Mon02] Richard Monson-Haefel. Enterprise JavaBeans, 3rd Edition. 2002. Sebastopol, CA. O´Reilly, 2002. ISBN: 3897211920

[mvc] Sun Microsystems, Inc.1995-2002. Designing Enterprise Applications with the J2EE™ Platform, Second Edition. http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/web-tier/web-tier5.html . Februar 2002

[OCo02] Daniel O´Connor. 2002. Inheritance & EJBs. www.theserverside.com/resources/article.jsp?l=EJBInheritance. Januar 2002.

[RoAmJe02] Ed Roman, Scott W. Ambler, Tyler Jewell. Mastering Enterprise JavaBeans, Second Edition . New York, NY .John Wiley & Sons Inc., 2002. ISBN: 0-471-41711-4

[servlet] Sun Microsystems, Inc. 1995-2002. JAVA™ SERVLET TECHNOLOGY - The Power Behind the Server. http://java.sun.com/products/servlets. Januar 2002

[struts] Apache Software Foundation. 1999-2002. Struts. http://jakarta.apache.org/struts/userGuide/index.html. Januar 2002

[strutscon] James Holmes. 2002. Struts Console http://www.jamesholmes.com/struts/console. April 2002

[Tomcat] Apache Software Foundation. 1999-2002. Apache Tomcat. http://jakarta.apache.org/tomcat/tomcat-4.1-doc/index.html. Januar 2002

[Sun02] Sun Microsystems, Inc. 2002. Enterprise JavaBeans Specification, Version 2.1. http://java.sun.com/products/ejb/docs.html, Januar 2002.

[wml] Ralf Peter Korte. April 2001. WML-Tutorial (Version 01.0422). http://www.wml-tutorial.de. Februar 2002

[XDoc] XDoclet Team. 2000-2002. Xdoclet – Attribute Oriented Programming Version 1.1.2. http://xdoclet.sourceforge.net/1.2beta/index.html. Mai 2002

[xml] 2002 W3C® (MIT, INRIA, Keio). 2002/02/17. Extensible Markup Language (XML). http://www.w3.org/XML. März 2002

/99/ T. Bühren, M. Cakir, E. Can, A. Dombrowski, G. Geist, V. Gruhn, M. Gürgrn, S. Handschumacher, M. Heller, C. Lüer, D. Peters, G. Vollmer, U. Wellen, J. von Werne
Endbericht der Projektgruppe eCCo (PG 315)
Electronic Commerce in der Versicherungsbranche
Beispielhafte Unterstützung verteilter Geschäftsprozesse
Februar 1999

/100/ A. Fronk, J. Pleumann,
Der DoDL-Compiler
August 1999

/101/ K. Alfert, E.-E. Doberkat, C. Kopka
Towards Constructing a Flexible Multimedia Environment for Teaching the History of Art
September 1999

/102/ E.-E. Doberkat
An Note on a Categorial Semantics for ER-Models
November 1999

/103/ Christoph Begall, Matthias Dorka, Adil Kassabi, Wilhelm Leibel, Sebastian Linz, Sascha Lüdecke, Andreas Schröder, Jens Schröder, Sebastian Schütte, Thomas Sparenberg, Christian Stücke, Martin Uebing, Klaus Alfert, Alexander Fronk, Ernst-Erich Doberkat
Abschlußbericht der Projektgruppe PG-HEU (326)
Oktober 1999

/104/ Corina Kopka
Ein Vorgehensmodell für die Entwicklung multimedialer Lernsysteme
März 2000

/105/ Stefan Austen, Wahid Bashirazad, Matthais Book, Traugott Dittmann, Bernhard Flechtker, Hassan Ghane, Stefan Göbel, Chris Haase, Christian Leifkes, Martin Mocker, Stefan Puls, Carsten Seidel, Volker Gruhn, Lothar Schöpe, Ursula Wellen
Zwischenbericht der Projektgruppe IPSI
April 2000

/106/ Ernst-Erich Doberkat
Die Hofzwerge — Ein kurzes Tutorium zur objektorientierten Modellierung
September 2000

/107/ Leonid Abelev, Carsten Brockmann, Pedro Calado, Michael Damatow, Michael Heinrichs, Oliver Kowalke, Daniel Link, Holger Lümkemann, Thorsten Niedzwetzki, Martin Otten, Michael Rittinghaus, Gerrit Rothmaier
Volker Gruhn, Ursula Wellen
Zwischenbericht der Projektgruppe Palermo
November 2000

/108/ Stefan Austen, Wahid Bashirazad, Matthais Book, Traugott Dittmann, Bernhard Flechtker, Hassan Ghane, Stefan Göbel, Chris Haase, Christian Leifkes, Martin Mocker, Stefan Puls, Carsten Seidel, Volker Gruhn, Lothar Schöpe, Ursula Wellen
Endbericht der Projektgruppe IPSI
Februar 2001

/109/ Leonid Abelev, Carsten Brockmann, Pedro Calado, Michael Damatow, Michael Heinrichs, Oliver Kowalke, Daniel Link, Holger Lümkemann, Thorsten Niedzwetzki, Martin Otten, Michael Rittinghaus, Gerrit Rothmaier
Volker Gruhn, Ursula Wellen
Zwischenbericht der Projektgruppe Palermo
Februar 2001

/110/ Eugenio G. Omodeo, Ernst-Erich Doberkat
Algebraic semantics of ER-models from the standpoint of map calculus.
Part I: Static view
März 2001

/111/ Ernst-Erich Doberkat
An Architecture for a System of Mobile Agents
März 2001

/112/ Corina Kopka, Ursula Wellen
Development of a Software Production Process Model for Multimedia CAL Systems by Applying Process Landscaping
April 2001

/113/ Ernst-Erich Doberkat
The Converse of a Probabilistic Relation
Oktober 2002

/114/ Ernst-Erich Doberkat, Eugenio G. Omodeo
Algebraic semantics of ER-models in the context of the calculus of relations.
Part II: Dynamic view
Juli 2001

/115/ Volker Gruhn, Lothar Schöpe (Eds.)
Unterstützung von verteilten Softwareentwicklungsprozessen durch integrierte Planungs-, Workflow- und Groupware-Ansätze
September 2001

/116/ Ernst-Erich Doberkat
The Demonic Product of Probabilistic Relations
September 2001

/117/ Klaus Alfert, Alexander Fronk, Frank Engelen
Experiences in 3-Dimensional Visualization of Java Class Relations
September 2001

/118/ Ernst-Erich Doberkat
The Hierarchical Refinement of Probabilistic Relations
November 2001

/119/ Markus Alvermann, Martin Ernst, Tamara Flatt, Urs Helmig, Thorsten Langer, Ingo Röpling,
Clemens Schäfer, Nikolai Schreier, Olga Shtern
Ursula Wellen, Dirk Peters, Volker Gruhn
Project Group Chairware Intermediate Report
November 2001

/120/ Volker Gruhn, Ursula Wellen
Autonomies in a Software Process Landscape
Januar 2002

/121/ Ernst-Erich Doberkat, Gregor Engels (Hrsg.)
Ergebnisbericht des Jahres 2001
des Projektes “MuSofT – Multimedia in der SoftwareTechnik”
Februrar 2002

/122/ Ernst-Erich Doberkat, Gregor Engels, Jan Hendrik Hausmann, Mark Lohmann, Christof Veltmann
Anforderungen an eine eLearning-Plattform — Innovation und Integration —
April 2002

/123/ Ernst-Erich Doberkat
Pipes and Filters: Modelling a Software Architecture Through Relations
Juni 2002

/124/ Volker Gruhn, Lothar Schöpe
Integration von Legacy-Systemen mit Eletronic Commerce Anwendungen
Juni 2002

/125/ Ernst-Erich Doberkat
A Remark on A. Edalat’s Paper *Semi-Pullbacks and Bisimulations in Categories of Markov-Processes*
Juli 2002

/126/ Alexander Fronk
Towards the algebraic analysis of hyperlink structures
August 2002

/127/ Markus Alvermann, Martin Ernst, Tamara Flatt, Urs Helmig, Thorsten Langer
Ingo Röpling, Clemens Schäfer, Nikolai Schreier, Olga Shtern
Ursula Wellen, Dirk Peters, Volker Gruhn
Project Group Chairware Final Report
August 2002

/128/ Timo Albert, Zahir Amiri, Dino Hasanbegovic, Narcisse Kemogne Kamdem,
Christian Kotthoff, Dennis Müller, Matthias Niggemeier, Andre Pavlenko, Stefan Pinschke,
Alireza Salemi, Bastian Schlich, Alexander Schmitz
Volker Gruhn, Lothar Schöpe, Ursula Wellen
Zwischenbericht der Projektgruppe Com42Bill (PG 411)
September 2002

/129/ Alexander Fronk
An Approach to Algebraic Semantics of Object-Oriented Languages
Oktober 2002

/130/ Ernst-Erich Doberkat
Semi-Pullbacks and Bisimulations in Categories of Stochastic Relations
November 2002

/131 Yalda Ariana, Oliver Effner, Marcel Gleis, Martin Krzysiak,
Jens Lauert, Thomas Louis, Carsten Röttgers, Kai Schwaighofer,
Martin Testrot, Uwe Ulrich, Xingguang Yuan
Prof. Dr. Volker Gruhn, Sami Beydeda
Endbericht der PG nightshift:
Dokumentation der verteilten Geschäftsprozesse im FBI und Umsetzung von Teilen dieser Prozesse im Rahmen eines FBI-Intranets basierend auf WAP- und Java-Technologie
Februar 2003