INSTITUT FÜR INFORMATIK
Lehr- und Forschungseinheit für
Programmierung und Softwaretechnik
Oettingenstraße 67 D–80538 München

Erweiterung des Uniworx-Systems für Bachelorstudiengänge

Georgi Dikov

Fortgeschrittenenpraktikum

Beginn der Arbeit:	01.12.2007
Abgabe der Arbeit:	28.05.2008
Betreuer:	Prof. Dr. Martin Wirsing
	Andreas Schroeder

Erklärung

Hiermit versichere ich, dass ich diese Ausarbeitung selbständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

München, den 28.05.2008 Georgi Dikov

Zusammenfassung

Im Wintersemester 2007-2008 wurde bei der Ludwig-Maximilians-Universität den Studiengang „Informatik Bachelor“ eingeführt. Da dieser Studiengang eine neue und für die Studenten noch unbekannte Studienordnung hat, entstand der Bedarf nach einem automatisierten Studiumswegweiser, der die Studenten bei der Wahl ihrer zu besuchenden Veranstaltungen unterstützt. An dem Institut für Informatik ist ein Softwaresystem zur Unterstützung des Übungsbetriebs bereits eingesetzt. In diesem System können die Studenten unter anderem ihre Übungsaufgaben abgeben, sich für Vorlesungen und Klausuren anmelden usw. Da dieses System Studenten und Übungsleitern bereits bekannt ist, lag es nah es um die den Bachelor-Studiengang unterstützenden Funktionalitäten zu erweitern, anstatt ein neues System zu entwickeln. In folgender Arbeit wird der Entwurf und die Implementierung dieser Funktionalitäten beschrieben.

Inhaltsverzeichnis

1 Einleitung

Derzeit werden an vielen deutschen Universitäten Studiengänge mit den Abschlüssen Bachelor und Master eingeführt. Die bisher bestehenden Diplom-Studiengänge Informatik und Medieninformatik werden daher, zum Wintersemester 2007/08 jeweils in einen Bachelor-/Master-Studiengang überführt.
Der wesentliche Unterschied zwischen Diplom- und Bachelorstudiengängen sind die studienbegleitenden Prüfungen in den Bachelor-Programmen: Jedes Modul (bestehend meistens aus Vorlesung und Übung) wird am Ende der Veranstaltung durch eine Klausur oder mündliche Prüfung geprüft. Für eine bestandene Prüfung gibt es eine Note und es werden so genannte ECTS-Punkte angerechnet. Insgesamt sind 180 ECTS Punkte zu erwerben, was, auf sechs Semester verteilt, 30 ECTS Punkte pro Semester ergibt. Es gibt Module mit 6, 9, 12 und 15 ECTS-Punkten, die zum Teil aus mehreren Einzelveranstaltungen bestehen([fI]). Um sein Studium erfolgreich abzuschließen, muss der Student alle nach seiner Studienordnung vorgesehenen Punkte fristgemäß sammeln und schließlich eine Bachelor-Arbeit schreiben. Zusätzlich gibt es eine Orientierungsprüfung, die spätestens bis zum Ende des dritten Semesters erfolgreich abgelegt werden muss.
Der Zeitdruck für die Bachelor-Studenten ist groß, da sie ihre 180 ECTS-Punkte in 6 Semester sammeln müssen. Damit sie das schaffen, ist es empfohlen, dass sie sich an dem in der Studienordnung vorgesehenen Studienplan halten. Da dieser Studienplan nicht einfach für jeden einzelnen Student zu erstellen ist, entstand der Bedarf nach einem automatisierten Wegweiser für diesen neuen Studiengang. Die Grundidee ist ein Softwaresystem bereit zu stellen, in dem jeder Student:

1. Sein Studiumsfortschritt nachverfolgen kann.
2. Seine im aktuellen Semester zu besuchenden Vorlesungen sehen kann.
3. Kritische Deadlines sehen kann.
4. Informationen über besuchte Veranstaltungen bekommen kann.

Sehr wichtig und hilfreich für die Studenten ist, dass sie immer wissen, was sie noch machen müssen und wie gut ihr Studium läuft. Deswegen sollen im System Informationen über ihre erbrachten Leistungen gespeichert sein. Da Daten wie Noten und besuchte Veranstaltungen vertrauliche Daten sind, stellt es sich eine hohe Sicherheitsanforderung an dem System an, damit die Datenschutzrichtlinien erfüllt werden. Das Tool muss auch sicherstellen, dass die Daten richtig sind, die im System eingetragen werden, damit kein Student wegen einem inkorrekten Studienplan eine Vorlesung nicht besucht.

Als zusätzliches Feature sollte das Tool auch sicherstellen, dass die Klausuren möglichst überschneidungsfrei sind, damit alle betroffenen Studenten an die Klausuren teilnehmen können. Die Klausurverwaltung sollte dadurch für die Assistenten auch einfacher sein.

Im Rahmen des Software-Entwicklungs-Praktikums im Sommersemester 2007 wurde die Aufgabe erteilt, ein solches System zu implementieren. Nach der Implementierung sollte es in den bereits laufenden Systemen des Instituts für Informatik an der LMU integriert werden, was die Aufgabe im Rahmen dieses Fortgeschrittenenpraktikums ist.

2 Uniworx 1.5

Uniworx ist ein Java-Struts ([Foua]) basiertes System. Ihre zentrale Funktionalität ist die Unterstützung des Übungsbetriebs an dem Institut für Informatik, indem es unter anderem den Studenten die Möglichkeit anbietet, dort ihre Hausaufgaben abzugeben oder sich für Klausuren anzumelden. In dem Uniworx-System werden auch die persönlichen Daten der Studenten erfasst, die von den Übungsleitern beim Verschicken von Übungsscheinen benutzt werden.

Das System wird auf einem Apache Tomcat Server ([Foub]) gehostet. Für die Abbildung der Java-Objekte in eine relationale Datenbank wird das Tool Hibernate ([CB06]) benutzt. Somit werden die Java-Objekten mit ihren Beziehungen und Eigenschaften in eine MySQL Datenbank ([Mic]) gespeichert. Hibernate übernimmt dabei das Object-Relational Mapping (siehe unten) komplett. Er erledigt sowohl die Erstellung des Datenbankschemas als auch das Lesen und Speichern der Java-Objekte. Durch Hibernateabfragen werden die Daten aus der Datenbank gelesen und automatisch Java-Objekte erstellt.

Uniworx ist mit Apache Struts 1.3.5 implementiert, ein Open-Source-Framework für die Entwicklung von Java-Webanwendungen. Struts wurde im Jahre 2000 von Craig McClanahan entwickelt und zählt heute zu einem der bekanntesten Jakarta-Projekte. Dem Struts-Framework liegt das Entwurfsmuster „Model View Controller“ (MVC) zugrunde. Das MVC ist ein Architekturmuster zur Strukturierung von Software-Komponenten in drei Gruppen: Datenmodell, Repräsentation und Programmsteuerung. Ziel der Architektur ist die logische Trennung der Software-Komponenten, damit die Entwicklung, Wartung und Erweiterung der Software vereinfacht wird.

Controller. Der Controller übernimmt die Interaktion mit dem Benutzer des Systems und leitet seine Anfragen an dem Modell in einer passender Form weiter.

Modell. In dem Modell sind die Datenobjekten und die Geschäftslogik untergebracht. Das Modell wird vom Controller angesprochen und erledigt die angeforderten Aufgaben, indem zum Beispiel einige Objekte ihre Zustand ändern oder verschiedene Daten abgefragt werden.

View. Die View-Komponenten sind für die Repräsentation der Daten zuständig und stellen die Datenobjekte aus dem Modell dar.

Das Apache Struts Framework ist eine Implementierung des Modell-View-Controller Entwurfsmusters. Die Steuerungskomponente werden durch Aktionen implementiert. Eine Aktion wird ausgeführt, wenn der Benutzer ein HTTP-Request auslöst. Die Aktionen kommunizieren mit dem Anwendungskern (das Modell) und leiten die von dem Benutzer gestellten Anfragen weiter. In dem Anwendungskern werden die Anfragen bearbeitet und das Ergebnis an den View-Komponenten geschickt, die es auf der Weboberfläche darstellen.

Das Request-processing bei Struts funktioniert nach folgendem Prinzip:

1. Der Benutzer initiiert einen HTTP-Request, in dem er auf einem Button oder einem Link in seinem Browser klickt.

2. Struts bietet ein Plugin mit sehr guten Validierungsmöglichkeiten an. Mit einer entsprechenden Einstellung in der zentralen Konfigurationsdatei von Struts struts-config.xml kann man die ganze Information, die in einem Formular übergeben wird, validieren, indem man festlegt, welche Felder wie gefüllt sein sollen.

3. Die Formulare sind ein sehr wichtiger Teil des Request-processings: ein Formularobjekt ist so zu sagen ein Container, wo die Eingabedaten für die Aktionen von dem Benutzer angegeben werden. Bei der Ausführung des HTTP-Requests werden die angegebenen Daten aus dem HTTP-Formular in eine Instanz der Formularklassen gespeichert und die Aktion mit diesem Formular als Eingabe ausgeführt. In struts-config.xml wird festgelegt, welche Aktion welches Formular als Eingabe bekommt.

4. Der HTTP-Request wird von einer Controller-Komponente aufgenommen und es wird eine Aktion ausgeführt. Die Aktionen sind die Komponente, die mit dem Anwendungskern kommunizieren.

5. Nach der Ausführung der Aktion werden die berechneten Ergebnisse in die so genannten JavaBeans zusammengefasst oder in der Session direkt gespeichert. In struts-config.xml wird festgelegt, wie die Request-Abarbeitung nach der Ausführung der Aktion ablaufen soll. Die Aktion liefert an dem RequestProcessor einen ActionForward-Wert und er führt die Operation aus, die in der Konfigurationsdatei angegeben ist (meistens ist da der Name einer View-Komponente angegeben).

6. Die entsprechende View-Komponente(JSP-Seite) wird geöffnet. Die JSP Seite greift auf die in der Session gespeicherten Daten zu und stellt sie im Webbrowser dar. Die JSP-Seiten benutzen neben HTML-Tags auch die von Struts gelieferten Taglibs, um die Daten aus den JavaBeans darzustellen. Man hat auch die Möglichkeit selbst JSP-Tags zu definieren. Das Ziel ist so wenig wie möglich Java-Code in den View-Komponenten zu haben, um höhere Wartbarkeit und Übersicht zu erreichen.

Eine detaillierte Beschreibung dieses Ablaufs wird anhand des Geschäftsvorfalls „Vorlesungsplan erstellen“ im Abschnitt 4.2 dargestellt.

Das Uniworx ist grundsätzlich in zwei Modulen Frontend und Backend unterteilt. In dem Frontend sind die Aktionen und die Formulare unterbracht. Die Aktionen werden von dem RequestProcessor gestartet und stellen die Verbindung zwischen der Weboberfläche und dem Anwendungskern dar. Sie benutzen bei ihrer Ausführung die Daten aus dem entsprechenden Formularobjekt (siehe oben). Die Formulare werden mit den Daten aus der HTTP-Formulare gefüllt und an die Aktionen übergeben. Durch die in den Formularen definierten Getter- und Settermethoden kann man auf die Daten zugreifen. Ein Formular kann auch vor der Ausführung der Aktion validiert werden.

Im Uniworx-System ist ein Teil der Businesslogik in den Aktionen implementiert. Nach dem MVC-Muster sollen die Aktionen nur Controller-Komponente sein, d.h. nur die Daten aus dem Anwendungskern anfordern.

In dem Backend sind die Data-Klassen, die Datenbankzugriffsklassen und die Applikation-Logik-Klassen untergebracht. Die Data-Klassen stellen die Entitäten dar, die die logischen Uniworx-Objekte repräsentieren. Wie oben erwähnt, werden die Data-Klassen in die Datenbank mit Hilfe des Tools Hibernate gespeichert. Die Datenbankzugriffsklassen benutzen eine Hibernate-Datenbank-Session, um Statements an die MySQL-Datenbank zu schicken. Das Ergebnis wird zu einer Data-Klasse gemappt. Das Object-Relational Mapping ist eine Technik, wie die Objekte der Anwendungsprogramm in eine relationale Datenbank gespeichert werden. Im einfachsten Fall werden Klassen auf Tabellen abgebildet, jedes Objekt entspricht einer Tabellenzeile und für jedes Attribut wird eine Tabellenspalte reserviert. Die Identität

eines Objektes entspricht dem Primärschlüssel der Tabelle. Hat ein Objekt eine Referenz auf ein anderes Objekt, so kann diese über eine Fremdschlüssel-Primärschlüssel-Beziehung in der Datenbank dargestellt werden. Wenn die Beziehungen zwischen den Objekten eine höhere Multiplizität (z.B. n-to-n) haben, muss eine zusätzliche Join-Tabelle angelegt werden.

In dem Package applicationLogik ist ein grosser Teil der Businesslogik implementiert. Die Klassen in diesem Modul bieten Funktionalitäten an, die von den Aktionen benutzt werden, um die Benutzeranfragen zu bearbeiten.

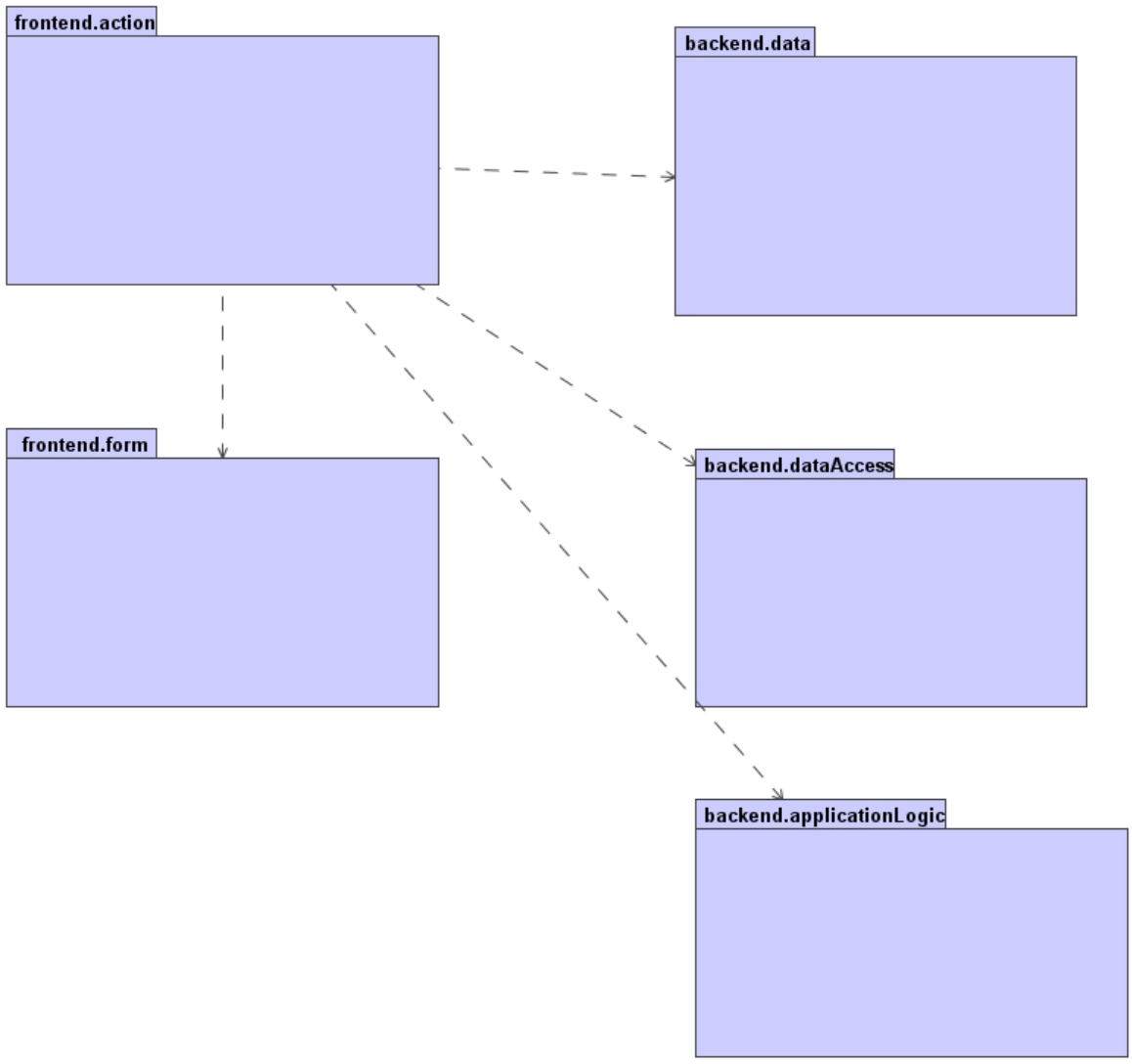

Abbildung 1: Uniworx Softwarearchitektur

3 Analyse

Bei der Entwicklung des neuen Systems sind wir nach dem Rational Unified Process (RUP)([GBR99]) vorgegangen. Bei dem RUP wird der Entwicklungsprozess in Iterationen unterteilt. Jede Itera-

tion umfasst wiederum sechs Disziplinen: Geschäftsprozessmodellierung (Business Modeling), Anforderungsanalyse (Requirements), Analyse und Design (Analysis and Design), Implementierung (Implementation), Test (Test), Auslieferung (Deployment). Der Anteil des Umfangs der einzelnen Disziplinen hängt von der Iteration ab.

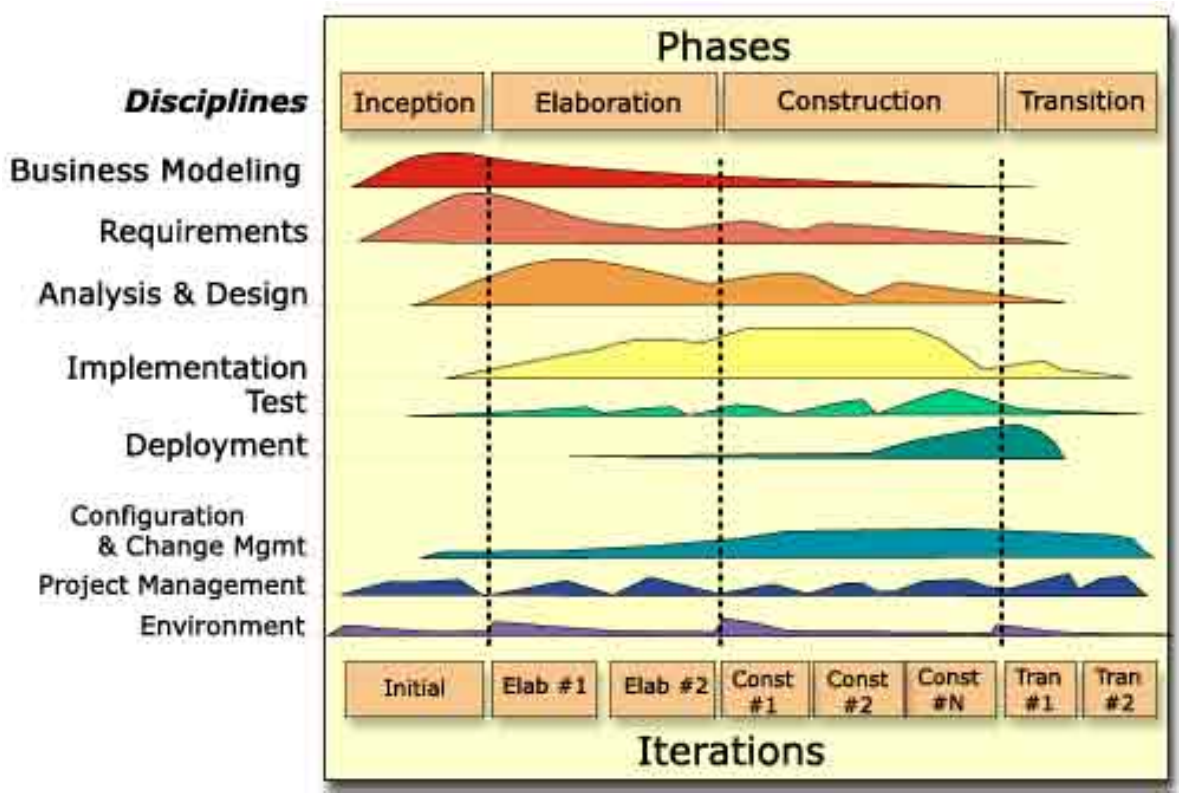

Abbildung 2: Rational Unified Process

Wir haben die Entwicklung in zwei Iterationen unterteilt. Bei der ersten Iteration haben wir uns auf die ersten drei Disziplinen (Geschäftsprozessmodellierung, Anforderungsanalyse und Analyse und Design) konzentriert und den Entwicklungs- und Testingsaufwand reduziert. Während der Geschäftsprozessmodellierung und der Anforderungsanalyse haben wir unsere Ideen gesammelt, indem sich das ganze Team zusammengesetzt hat und ein Brainstorming gemacht hat. Nachdem wir sieben Geschäftsvorfälle definiert haben (siehe 3.1) haben wir auch folgende weitere Aspekte und Punkte analysiert:

1. **Datenschutzrichtlinie.** Da in dem System vertrauliche Daten der Studenten wie Noten und besuchte Veranstaltungen gespeichert werden sollen, soll das System diese Daten vor unerwünschten Zugriffe schützen können.

2. **Benutzerverwaltung.** Jeder Student, der Bachelor Informatik an der LMU studiert, wird eine Kennung für das neue System brauchen.

3. **Einpflegen der Daten.** Die Hauptfunktionalität des Systems soll die Unterstützung der Studenten bei der Wahl ihrer zu besuchenden Veranstaltungen sein. Deshalb müssen die an dem Institute für Informatik angebotenen Veranstaltungen im System eingepflegt

werden. Das System muss auch die erbrachten Leistungen der Studenten in den besuchten Veranstaltungen wissen, damit es entscheiden kann, ob der Student die Veranstaltung nochmal besuchen muss.

4. **Spezifische Anforderungen des Bachelor Studiengangs.** Um den Studenten Vorschläge über die zu besuchenden Veranstaltungen zu machen, muss das System über Informationen aus der Studienordnung des Bachelor-Studiengangs verfügen. D.h. das System muss wissen welche Veranstaltung von welchen Studenten in welchem Semester belegt werden muss.

5. **Bereits vorhandene Systeme und Webseiten an dem Institut für Informatik.** Die Einführung eines neuen Systems ist kompliziert und mit viel Administrationsaufwand wie z.B. Benutzerverwaltung verbunden. Deswegen haben wir die bereits vorhandenen Systemen analysiert, um herauszufinden, ob es nicht günstiger ist, ein System zu erweitern, statt ein neues zu implementieren.

Nachdem wir die oben genannten Punkte analysiert haben, haben wir uns entschieden, dass kein neues separates System entwickelt, sondern ein schon bestehendes erweitern werden soll. Für die gestellten Anforderungen und Ziele passte das System „Uniworx“ sehr gut, welches den Übungsbetrieb an dem Institut für Informatik verwaltet. Die Vorteile des Uniworx-Systems waren, dass fast alle Studenten, die an der LMU Informatik studieren, das System schon kennen und eine Kennung dafür haben. Dadurch entfiel die notwendige Benutzerverwaltung. Die Pflege des Systems wird auch einfacher sein, da die Studenten und Assistenten ihre Daten nur in einem System und nicht in zwei pflegen müssen, was zu Redundanz der Daten führen würde. Zusätzlich sind die Datenschutzrichtlinien durch Uniworx bereits erfüllt. Eine detaillierte Beschreibung des Uniworx steht im Abschnitt 2.

3.1 Geschäftsvorfälle

Nach der ersten Analyse der Systemanforderungen haben wir sieben Geschäftsvorfälle definiert. Ein Geschäftsvorfall ist ein Arbeitsprozess, der von einer Organisation erbracht wird (z.B. eine Reisebüro bucht für einen Kunden eine Reise). Der Geschäftsvorfall besteht aus vielen Einzelschritten, die automatisiert oder manuell von verschiedenen Systemen/Aktoren erbracht werden. Er überschreitet die Systemgrenzen und umfasst einen oder mehrere Anwendungsfälle. Folgende Geschäftsvorfälle sind für das zu entwickelnde System definiert:

1. **Geschäftsvorfall „Studiumsüberblick erstellen“.** Ziel dieses Geschäftsvorfalls ist, dass der Student einen Überblick über alle bereits besuchten und noch zu besuchende Vorlesungen bekommt. Er kann seinen Studiumsfortschritt nachverfolgen.

2. **Geschäftsvorfall „Vorlesungsplan erstellen“.** Das System soll dem Studenten einen Vorschlag machen, welche Vorlesungen er im aktuellen Semester besuchen muss. Das sind alle Vorlesungen, die laut dem Studienplan für das jeweilige Semester geplant sind und alle Vorlesungen aus den früheren Semestern, wo der Student durchgefallen ist.

3. **Geschäftsvorfall „Studienordnung ändern“.** Dieser Geschäftsvorfall beschäftigt sich mit dem Einspielen einer Studienordung in das System. In der Studienordnung sollen alle Informationen, die für den Geschäftsvorfall „Vorlesungsplan erstellen“ benötigt

werden, wie z.B. welche Veranstaltung in welchem Semester zu besuchen ist, vorhanden sein.

4. **Geschäftsvorfall „Statistik anzeigen“.** Ziel dieses Geschäftsvorfall ist, dass ein Assistent die Möglichkeit hat, eine Statistik über die Klausur einer von ihm betreuten Vorlesung graphisch einsehen zu können.

5. **Geschäftsvorfall „Ergebnisse verwalten“.** Dieser Geschäftsvorfall hat den Zweck, dass die Studenten ihre Klausurergebnisse selbst im System eintragen können, wenn sie nicht von einem Assistenten gepflegt werden. Das System soll dabei sicherstellen, dass die Noten, die von den Assistenten eingetragen werden, nicht überschrieben werden dürfen.

6. **Geschäftsvorfall „Prüfungstermin festlegen“.** Der Zweck dieses Geschäftsvorfalls ist, dass die Klausuren möglichst überschneidungsfrei stattfinden, damit alle betroffenen Studenten an die Klausuren teilnehmen können. Dafür soll der Assistent beim Planen einer Klausur die Möglichkeit haben, alle bereits eingetragenen Klausuren zu sehen, um entscheiden zu können an welchem Tag er seine Klausur anlegen möchte.

7. **Geschäftsvorfall „Studenten erinnern“.** Dieser Geschäftsvorfall dient dazu, dass die Studenten an kritischen Deadlines wie z.B. Klausuranmeldungsfristen, per email informiert werden.

3.2 Anwendungsfälle

Nach der Geschäftsvorfallanalyse haben wir insgesamt neun Anwendungsfälle (siehe Abbildung 3) definiert. Ein Anwendungsfall definiert eine Interaktion zwischen Akteuren und dem betrachteten System, die stattfindet, um ein bestimmtes fachliches Ziel zu erreichen. Der Anwendungsfall(auch Use Case genannt) wird unterbrechungsfrei ausgeführt und beschreibt die Vorgänge innerhalb der Systemgrenze. Die Use Cases und die Geschäftsvorfälle werden erfasst, indem man die Spezifikation analysiert und fachliche Interviews durchführt. Sie definieren den Funktionsumfang und -fähigkeiten des Systems. Die Geschäftsvorfälle beschreiben die Arbeitsprozesse allgemein und müssen nicht unbedingt mit dem betrachteten System verbunden sein. Die Anwendungsfälle beschreiben, was die verschiedenen Akteure mit dem System machen können, um ihre Geschäftsvorfälle zu erledigen. Folgende Use Cases sind für das neue System definiert:

1. **Anwendungsfall „Studiumsüberblick generieren“.** Ein Student bekommt einen Überblick über den Verlauf seines Studiums. Er sieht welche Veranstaltungen er besucht hat und welche Noten er bekommen hat. Er bekommt auch einen Überblick über die Veranstaltungen, die er in den nächsten Semestern besuchen muss.

2. **Anwendungsfall „Vorlesungsplan erstellen“.** Ein Student, der sich für eine Vorlesung anmelden möchte, bekommt einen Vorschlag vom System, welche Veranstaltungen für ihn relevant sind.

3. **Anwendungsfall „Studienordnung hinzufügen“.** Der Systemadministrator kann eine neue Studienordnung in das System einspielen.

4. **Anwendungsfall „Statistik anzeigen“.** Ein Assistent kann sich eine Statistik über die Ergebnisse einer Klausur, die zu einer von ihm betreutet Vorlesung gehört, erstellen lassen.

5. **Anwendungsfall „Ergebnisse angeben“.** Ein Student kann seine Ergebnisse zu einer Klausur selbst eintragen, wenn die entsprechende Veranstaltung so angelegt ist, dass sie dies erlaubt, und wenn der Assistent noch keine Ergebnisse eingetragen hat.

6. **Anwendungsfall „Prüfungstermine anzeigen“.** Ein Assistent oder ein Student kann sich alle Prüfungstermine anzeigen lassen. Der Student bekommt nur die Termine zu sehen, zu deren Vorlesungen er sich bereits angemeldet hat. Der Assistent sieht dagegen alle Prüfungstermine.

7. **Anwendungsfall „Prüfungstermin anlegen“.** Ein Assistent legt zu einer Vorlesung eine Klausur mit einem Prüfungstermin an.

8. **Anwendungsfall „Studenten informieren“.** Die Studenten werden vom System über kritischen Ereignisse(wie Klausuranmeldungsfristen) automatisch informiert.

9. **Anwendungsfall „Tägliche Prüfung“.** Das System führt in regelmäßigen Abständen eine Überprüfung aus, um festzustellen, ob irgendwelche Fristen bald ablaufen.

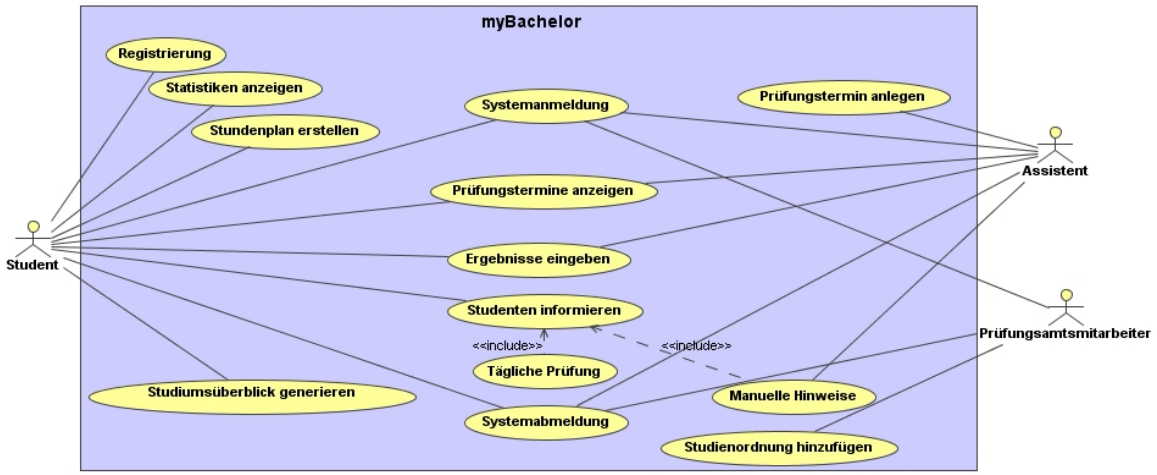

Abbildung 3: Use Case Diagramm

Als Beispiel wie ein Anwendungsfall definiert werden muss, ist die detaillierte Beschreibung des Use Cases „Vorlesungsplan erstellen“ dargestellt (siehe Abbildung 3.2).

Use Case Name	„Vorlesungsplan erstellen“
Kurzbeschreibung	Wenn sich der Student zu einer Vorlesung anmelden will, dann werden alle Vorlesungen, die laut der Studienordnung für das jeweilige Semester empfohlen sind, grün markiert. Zusätzlich werden auch die Veranstaltungen, die der Student in den früheren Semestern besuchen musste und nicht bestanden hat, grün markiert. Alle weiteren Veranstaltungen werden normal dargestellt.
Auslöser	Der Student wählt Funktion „zu Vorlesungen anmelden“
Vorbedingung	Student ist authentisiert
Ergebnis	Der Student siehr welche Veranstaltungen für ihn für das jeweilige Semester relevant sind.
Nachbedingung	Der Student ist zu einer Vorlesung angemeldet.
Beteiligte Akteure	Student
Standardablauf	1. Vorlesungsvorschlag generieren 2. Vorlesung auswählen 3. Vorlesungsanmeldung speichern
Varianten	a) Benutzerabbruch 2. Abbruch durch Benutzer 3. Vorlesungsanmeldung wird nicht gespeichert
Referenzen	-
Anmerkungen, Fragen	

4 Erste Iteration

Wir haben uns entschieden, die Entwicklung unseres Systems in zwei Iterationen abzuschließen. In der ersten Iteration sollten die zwei Use Cases „Studiumsüberblick generieren“ und „Stundenplan erstellen“ implementiert werden. Die restlichen Anwendungsfälle sollten in der zweiten Iteration implementiert werden.

4.1 Analyse

Die erste Iteration fängt mit der Analyse des aktuellen Uniworx-Domänenmodells und mit dem Entwurf eines neuen Domänenmodells an, welches auch unsere neuen Anforderungen erfüllen sollte. Für die zwei Use Cases, die in der ersten Iteration implementiert werden müssten, brauchten wir als erstes eine passende Darstellung eines Studienplans in unserem System. Ein Studienplan besteht aus Semestern, Modulen und Vorlesungen. In einem Semester muss der Student eine durch die Studienordnung festgelegte Menge an Modulen abschließen. Ein Modul besteht aus einer oder mehreren Vorlesungen, Übungen, Praktika, Seminaren, Bachelor-Arbeit oder Bachelor-Prüfung. Ein Beispiel-Studienplan sieht so aus:

1. Semester 24 ECTS Punkte
P Einfuhrung in die Programmierung (Java) — 9 ECTS Punkte
P Analysis fur Informatiker und Statistiker — 9 ECTS Punkte
P Lineare Algebra fur Informatiker —- 6 ECTS Punkte

In diesem Beispiel muss der Student im ersten Semester diese drei Module bestehen. Das Modul „Einführung in die Programmierung (Java)“ ist ein Pflichtmodul(P) und besteht aus der Vorlesung „Vorlesung Einführung in die Programmierung“ mit 4 ECTS Punkten und aus der Übung „Übung zu Einführung in die Programmierung“ mit 2 ECTS Punkten. Anhand eines ähnlichen Studienplans, der im System gespeichert ist und dem Studenten zugeordnet ist, muss das System einen Vorschlag für den Studenten generieren, welche Vorlesungen er in dem jeweiligen Semester besuchen muss.

Die Lehrveranstaltungen werden jährlich angeboten. Ein Student muss dabei nur einmal eine Veranstaltung durch erfolgreiche Teilnahme an der dazugehörigen Klausur bestehen. D.h. eine konkrete Vorlesung ist eine Instanz der in der Studienordnung vorgeschriebenen Vorlesung. Zum Beispiel die Vorlesung „Einführung in die Programmierung“ im Wintersemester 2007/08 ist eine konkrete Instanz der Vorlesung „Vorlesung Einführung in die Programmierung“, die zu dem Modul „Einführung in die Programmierung“ gehört. Um diesen Zusammenhang in dem System darzustellen haben wir zwei Ebenen von Objekten definiert: eine abstrakte und eine konkrete Ebene. Auf der abstrakten Ebene werden die Course angelegt. Der Course stellt eine Lehrveranstaltung dar, wie sie in der Studienordnung beschrieben ist. Zu einem Course gehören wichtige Informationen, wie ECTS Punkte, Klausurart(bewerten mit Note, mit Punkten oder mit „ja“ und „nein“) usw. Der Course wird mit einem Modul verlinkt, welches wiederum zu einem Semester gehört.

Die konkrete Instanz eines Courses ist ein Exercise. Die konkreten Instanzen der Vorlesungen können dieselben für unterschiedliche Studienordnungen sein. Zum Beispiel die konkrete Vorlesung „Einführung in die Programmierung“ wird von Studenten unterschiedlicher Studiengänge besucht, die nach unterschiedlichen Studienordnungen studieren. Für jeden Studiengang bzw. Studienordnung hat die Vorlesung andere Bedeutung und Voraussetzungen.

Deswegen wird ein Course zu mehreren Modulen verlinkt, damit eine und dieselbe Vorlesung unterschiedliche Eigenschaften abhängig von dem zugehörigen Modul besitzen kann. Durch das Trennen der Objekten auf zwei unterschiedlichen Ebenen ist es gewährleistet, dass das System auch für andere Studiengänge bzw. Studienordnungen außer Bachelor Informatik benutzt werden kann. Das finale Domänenmodell von Uniworx ist in Abbildung 4 und in Abbildung 5 dargestellt.

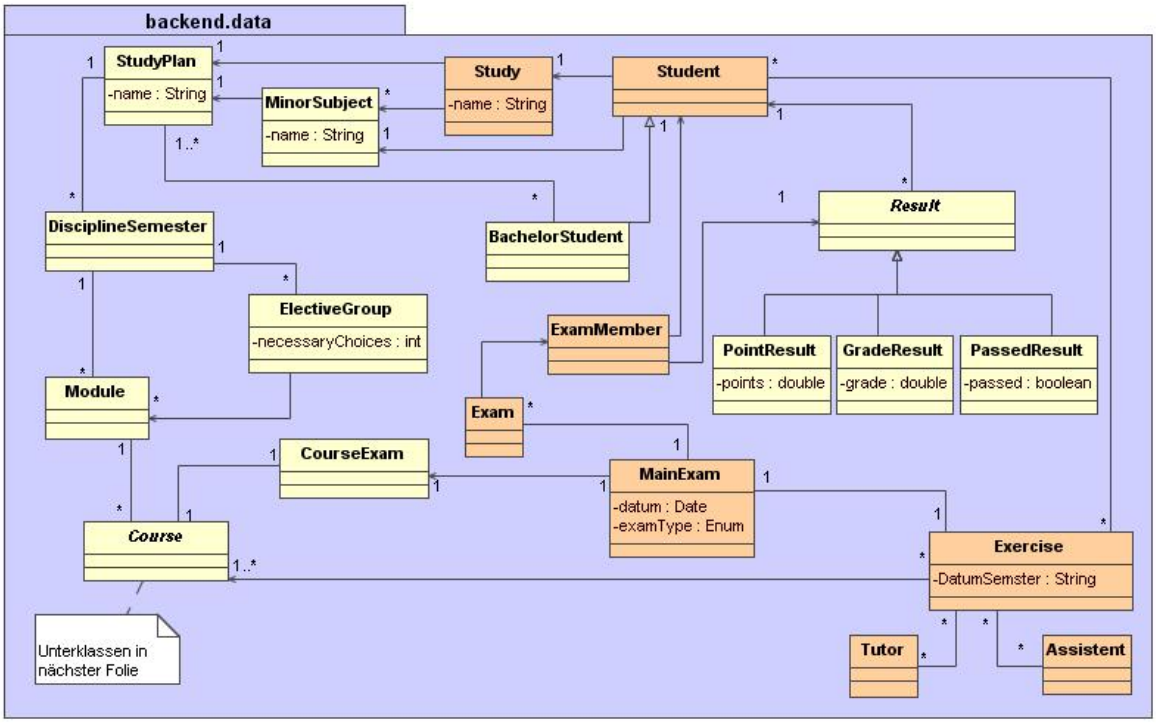

Abbildung 4: Uniworx Datenmodell

4.2 Entwurf

Der Anwendungsfall „Vorlesungsplan erstellen“ stellt die Grundfunktionalität des erweiterten Systems dar. Da im Uniworx-System die Studenten die Möglichkeit bereits hatten, sich für Vorlesungen anzumelden, haben wir uns gedacht, dass die zwei Use Cases „Vorlesungsplan erstellen“ und „Zu Vorlesung anmelden“ in einander integriert werden können. Der Student soll dabei vor der Anmeldung alle Vorlesungen, die laut der Studienordnung empfohlen sind, grün markiert bekommen. Im Folgenden wird detaillierter beschrieben, wie das „technisch“ realisiert ist.

4.3 Implementierung „Vorlesungsplan erstellen“

Der Anwendungsfall „Vorlesungsplan erstellen“ wird von einem Studenten ausgelöst, indem er in seinem Browser den Knopf „zur Vorlesungen anmelden“ klickt. Die HTTP-Aktion, die dadurch ausgelöst wird, heißt „signOnToExercise1.do“. Sie wird über einen HTTP-Request

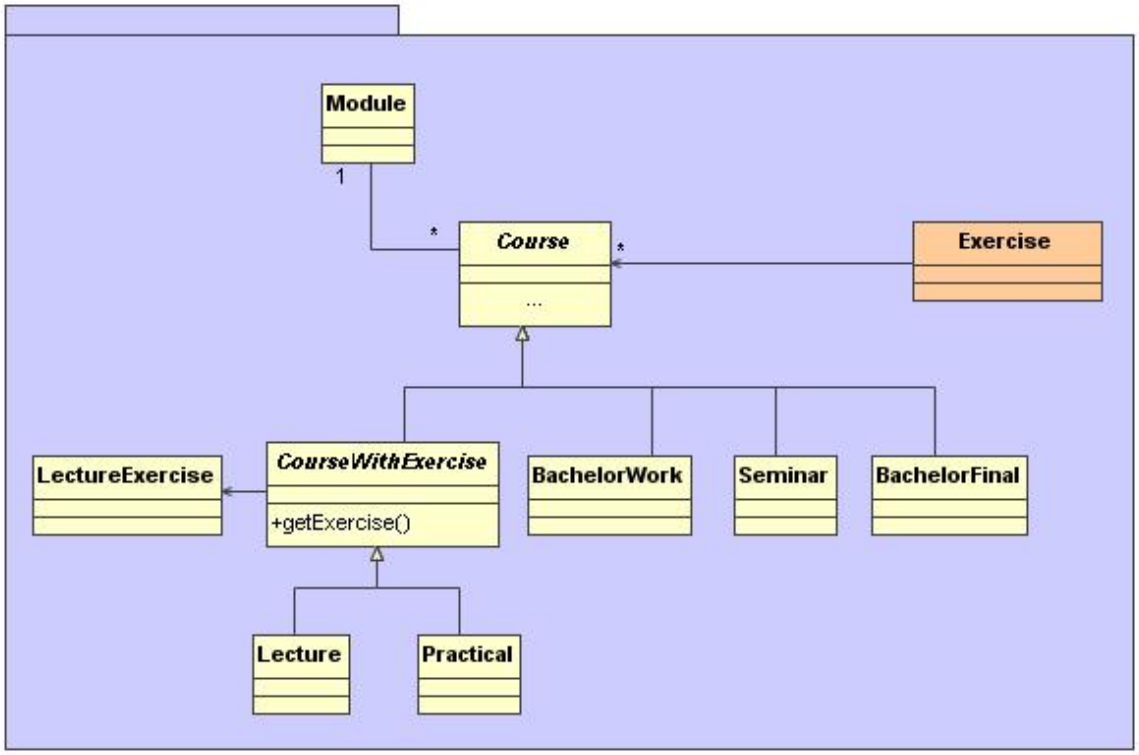

Abbildung 5: Unterteilung von Course

an dem Uniworx-RequestProcessor übergeben. Beim Starten des Uniworx-Systems liest der RequestProcessor alle Einträge aus der zentralen Konfigurationsdatei struts-config.xml, damit er beim Eintritt einer konkreten Aktion sie entsprechend bearbeiten kann. Der xml-Eintrag in der Konfigurationsdatei für die Aktion „signOnToExercise1“ sieht so aus:

```
<action path="/signOnToExercise1"
type="frontend.action.student.SignOnToExerciseAction1"
  scope="request" roles="Student">
  <forward name="loadEx" path="/student/signOnExercise1.jsp" />
  <forward name="noExerciseExists" path="/student/signOnExercise2.jsp" />
  <forward name="error" path="/error.jsp" />
</action>
```

Der Name „signOnToExercise1“ der HTTP-Aktion wird mit der Java Klasse frontend.action.student.SignOnToErxerciseAction1 gematcht und die Aktion wird ausgeführt. Sie erbt von der Oberklasse SecureAction. Die wichtigste Eigenschaft dieser Klasse ist die Methode delegate:

```
public ActionForward delegate(ActionMapping mapping, ActionForm form,
HttpServletRequest request, HttpServletResponse response, Session session)
```

Durch das ActionMapping und das ActionForward steuert die Aktion, wie die Navigation der Webseite nach ihrer Ausführung weiterläuft. Die ActionForm-Klasse stellt den HTML-

Formular dar, der von dem User gefüllt wird. In unserem Fall wird kein Formular benötigt, da keine Angaben durch den Studenten notwendig sind. Jede Uniworx Aktionsklasse muss von der SecureAction erben und dadurch einen gültigen Request-, Response und Sessionobjekt übernehmen. Die SecureAction ist eine Unterklasse der von Struts gelieferten Klasse Action, die noch zusätzlich immer prüft, ob der User des Systems angemeldet ist.

Die delegate-Methode wird automatisch gestartet. Die Methode holt sich an erster Stelle alle Vorlesungen, die in diesem Semester aktuell angeboten sind. Dafür wird die Datenbankzugriffsklasse backend.dataAccess.DAExercises angesprochen. Die dataAccess-Klassen sind für die Lese-Zugriffe zu der Datenbank gedacht. Sie benutzen die Hibernate Bibliotheken für Datenbankabfragen und für Session- und Transaktionssteuerung. Das Ergebnis der Abfrage wird sofort zu Java-Objekten gemappt. In unserem Fall ist der Rückgabewert von DAExercises eine Liste von Exercises.
Nachdem alle aktuell angebotenen Exercises geladen sind, werden diejenigen ausgefiltert, zu welchen der Student schon angemeldet ist. Falls die verbliebene Liste leer ist, wird ein „noExerciseExists“-Forwarding zurückgegeben, ansonst wird die Methode calculateProposal() gestartet. Diese Methode soll die Vorlesungen aussuchen, die für den Studenten laut seiner Studienordnung relevant sind und von ihm noch nicht bestanden sind. Als Eingabeparameter bekommt die Methode den HttpServletRequest und die Liste der Vorlesungen, zu denen der Student nicht angemeldet ist. Die Information über eine konkrete Vorlesung wird in dem Objekt Course gespeichert(siehe Abschnitt 4.1). Da ein Course viele Module haben kann, wird die Methode getModuleForStudyPlan von der Klasse Course benutzt, die für ein bestimmtes Course das passende Modul in dem Studienplan findet. Nachdem das richtige Modul-Objekt geladen ist, wird geprüft, ob der Course für den Studenten erforderlich ist und ob er zu einem Wahlpflichtmodul gehört, d.h. der Student darf von zwei oder mehreren Coursen nur einen auswählen. In diesem Fall werden alle Course gelesen, die zu dem Wahlpflichtmodul gehören, und wird geprüft, ob der Student einen von denen schon bestanden hat. Falls das nicht der Fall ist, wird die Vorlesung zu der Menge der empfohlenen Vorlesungen hinzugefügt. Wenn der Course für den Studenten erforderlich ist und nicht zu einem Wahlpflichtmodul gehört, dann wird nur geprüft, ob der Student ihn schon bestanden hat.

Alle Veranstaltungen, die als empfohlen bewertet sind, werden in der Session in zwei Gruppen gesetzt: eine für die Wahlpflicht-Veranstaltungen und eine für die weiteren empfohlenen Veranstaltungen. Nach einer erfolgreichen Ausführung der Aktion wird das ActionMapping-Forward wird auf „loadEx“ gesetzt. Somit leitet der RequestProcessor die Programmausführung zu der jsp-Seite signOnExercise1.jsp weiter.

5 Zweite Iteration

Während der zweiten Iteration haben wir die restlichen Use Cases im System implementiert. Der wichtigste Anwendungsfall für die angestrebten Anforderungen war der Fall „Studienordnung hinzufügen“, da dadurch viele für den Anwendungsfall „Vorlesungsplan erstellen“ notwendigen Informationen im System gespeichert werden. Um eine Studienordnung mit allen Informationen über die Lehrveranstaltungen, deren Abhängigkeiten und Zuordnung zu Semestern in einer strukturierten und für das Uniworx-System verständlichen Form zu bekommen, haben wir den Java-Compiler-Compiler(JavaCC)([Jav]) benutzt. Wir haben einen Parser geschrieben, der eine in einer passenden Form geschriebene Studienordnung lesen und

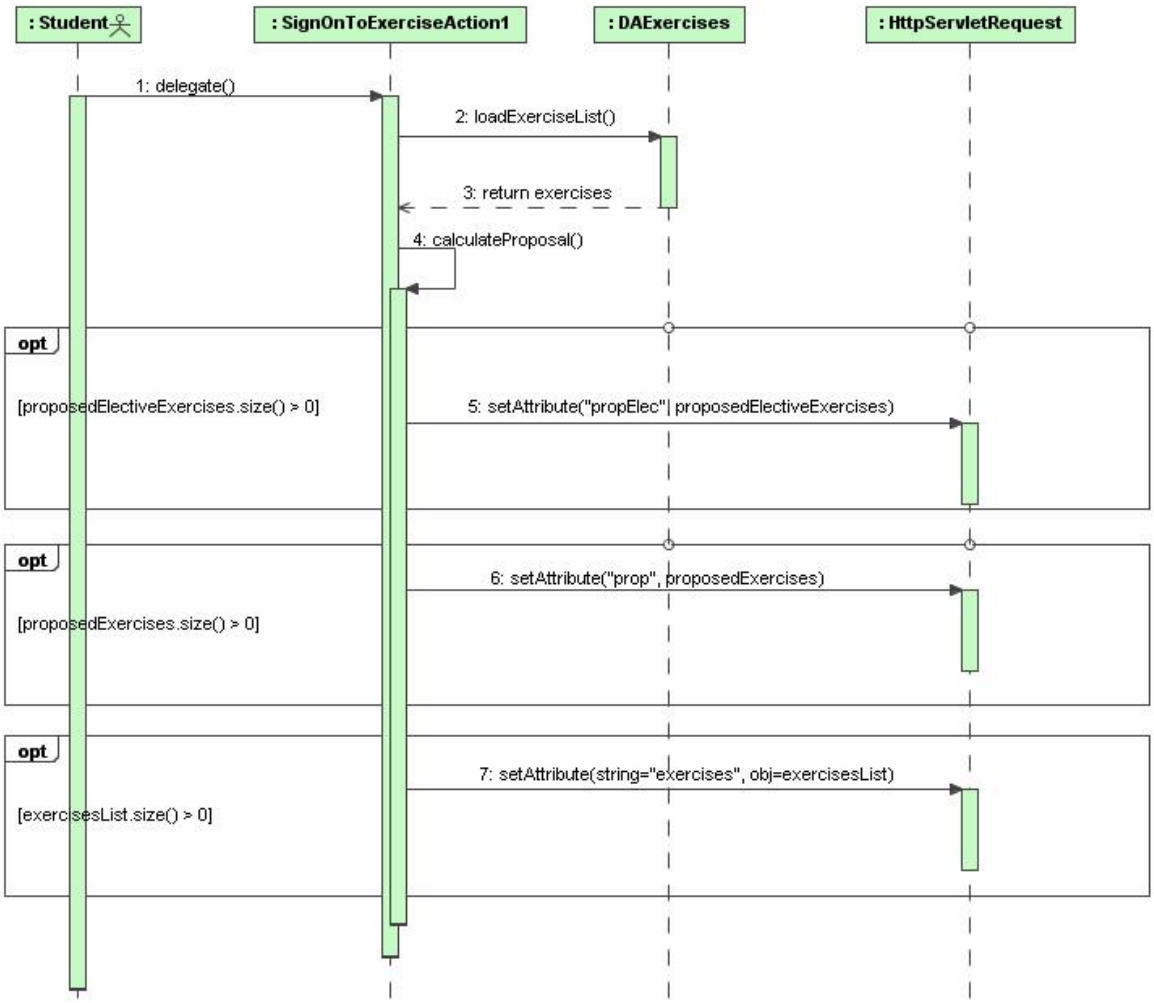

Abbildung 6: Interaktionsdiagramm "Vorlesungsplan erstellen"

in die Datenbank speichern kann. Eine Beispielstudienordnung sieht so aus:

```
Studienplan( Informatik Bachelor | hauptfach | 6){
Fachsemester(1){
Modul(Einfuehrung in die Programmierung | Inhalt: Dieses Modul
 gibt eine Einfuehrung in die ... | true| 12| 6){
Vorlesung(Vorlesung Einfuehrung in die Programmierung |
Einfuehrung in die Programmierung | Inhalt: Die folgenden
Themen werden ... | 6 | 4 | keine){
Klausur(Einfuehrung in die Programmierung |  true)
Uebung(Uebung Einfuehrung in die Programmierung | Inhalt: Die
in der Vorlesung besprochenen ... | 3 | 2)
}
}
...
```

```
}
}
```

Durch die in JavaCC definierten Regeln kann der Parser die unterschiedlichen Elemente erkennen und sie entsprechend behandeln. Die Regeln sind nach folgendem Prinzip aufgebaut: sie suchen nach bestimmten Wörtern bzw. Kombinationen von Wörtern und Zeichen. Zum Beispiel die Regel für Modul fängt mit dem Schlüsselwort „Modul“ an, gefolgt von Klammern. In den Klammern werden die Parameter des Moduls eingegeben, wie zum Beispiel Name, Beschreibung, ECTS Punkte usw. Falls das gelesene Element Unterelemente hat(zum Beispiel ein Modul hat eine oder mehrere Vorlesungen; ein Semester hat viele Module usw.), werden die Unterobjekte innerhalb eines mit geschleiften Klammern erfassten Blocks definiert.

Sobald der Parser ein Schlüsselwort findet, das zu einer Regel passt, wendet er die entsprechende Regel an. In der Regel selbst ist Java-Code integriert, welcher eine Verbindung zu der Datenbank aufbaut, ein Objekt der entsprechenden Uniworx-Dataklasse erstellt(z.B. Course, Module, StudyPlan usw.) und es in der Datenbank speichert. Somit kann man leicht neue Studienpläne ins System einspielen und das System kann leicht weitere Studiengänge aufnehmen.

Die Hierarchie der Studienordnungsgrammatik sieht so aus:

1. Studienplan
    - Fachsemester
        - Modul
            - Vorlesung
                - Klausur
                - Übung
            - Seminar
                - Klausur
            - Praktikum
                - Klausur
                - Übung
            - Bachelorarbeit
                - Klausur
            - Bachelorprüfung
                - Klausur

6 Zusammenführung Uniworx 1.5 und Uniworx 2.0

In Rahmen eines Fortgeschrittenenpraktikums sollen die zwei Systeme zu einem zusammengeführt werden und entsprechende Tests geschrieben werden. Das Hauptziel während der Integration ist, dass alle Funktionalitäten des alten Systems und alle Fortschritte, die parallel zu der Entwicklung des Neuen geschaffen sind, beibehalten werden. Deswegen sollen alle Änderungen im Code manuell nachgefügt und überprüft werden. Wichtig ist, dass alle

Studenten, die nach der alten Diplom-Informatik-Studienordnung studieren, von den neuen Funktionalitäten nicht betroffen sind und ihre alten Aufgaben mit dem Uniworx-System immer noch erledigen können.

Die wesentliche Weiterentwicklung im Uniworx 1.5 ist die Datenbank-Sessionverwaltung. Früher hatte ein Benutzer für seine ganze Sitzung nur eine Datenbanksession, wo alle gelesenen Objekte gespeichert wurden. Das könnte zu einem Dirty-Read führen, da man auf Objekte zugreift, die in der Zwischenzeit von einem anderen Benutzer modifiziert werden können. Z.B. wenn der Assistent etwas an einer Vorlesung ändert und der Student sie schon in seiner Session geladen hat, dann sieht er die Änderungen nicht und benutzt die inkonsistenten Daten weiter. Jetzt wird bei jeder Abfrage eine neue Datenbanksession erzeugt, die nach ihrer Ausführung geschlossen wird. So ist es sichergestellt, dass alle Daten konsistent sind.

Für die Use Cases „Studiumsüberblick generieren“ und „Vorlesungsplan erstellen“ sind auch Test Cases geschrieben. Für die Tests habe ich die Struts-Bibliotheken strutstest-2.1.4 ([Sou]) benutzt. Mit Hilfe dieser Bibliothek simuliert man die komplette Ausführung einer Aktion, indem die gleiche Umgebung wie bei der Ausführung auf dem Tomcat Server erstellt wird. Dafür werden die von der Aktion benötigten Eingabeparameter in dem Request gesetzt und erst dann die Aktion ausgeführt. Nach der Ausführung werden der Forward-Wert und die durch die Aktion gesetzten Sessionparameter geprüft. Wenn die resultierenden Werte mit den erwarteten übereinstimmen, heißt es, dass die Aktion ihre Aufgabe korrekt erfüllt hat und der Test „bestanden“ ist. Der Vorteil von den Struts Tests Cases ist, dass sie unabhängig von einem Server ausgeführt werden können. D.h. man braucht die Applikation nicht auf dem Server zu laden und kann nur lokal ein bestimmten Ablauf testen. Die komplette Umgebung für die Ausführung der Aktion wird simuliert und man kann auch leicht mit modifizierten Daten herumprobieren.

7 Zusammenfassung

Mit der Einführung von Studiengängen mit den Abschlüssen Bachelor und Master an der Ludwig-Maximilians-Universität entstand der Bedarf von einem automatisierten Studiumswegweiser, der die Bachelor- und Master-Studenten bei der Wahl ihrer zu besuchenden Veranstaltungen unterstützen soll. Durch das Software Entwicklungspraktikum im Sommer-Semester 2007 und das Fortgeschrittenenpraktikum im Winter-Semester 2007-2008 ist ein System entwickelt worden, das seine ursprünglichen Ziele und Anforderungen soweit erfüllt. Das Hauptziel, nämlich die Unterstützung des Bachelor-Informatik-Studiengangs durch einen Studienplanwegweiser, ist erfüllt und kann leicht und einfach von den betroffenen Studenten benutzt werden. Das System bietet noch weitere Funktionalitäten an, wie die graphische Anzeige des Studiumsfortschritts eines Studenten und eine vereinfachte Klausurenverwaltung. Das System steht für Weiterentwicklung noch offen und kann auch für weitere Studiengänge angesetzt werden.

Literatur

[CB06] Gavin King Christian Bauer. *Java Persistence with Hibernate*. Manning, 2006.

[fI] Institut für Informatik. *Studienordnung Informatik Bachelor*. Institut für Informatik, http://www.ifi.lmu.de/Studium/Studiengaenge/Informatik/InformatikBA/index.html.

[Foua] Apache Software Foundation. *Apache Struts Homepage*. Apache Software Foundation, http://struts.apache.org/.

[Foub] Apache Software Foundation. *Apache Tomcat Server Homepage*. Apache Software Foundation, http://tomcat.apache.org/.

[GBR99] Ivar Jacobson Grady Booch and James Rumbaugh. *The Unified Software Development Process*. Addison Wesley, 1999.

[Jav] Java.net. *Java Compiler Compiler*. Java.net, https://javacc.dev.java.net/.

[Mic] Sun Microsystems. *MySQL Database*. Sun Microsystems, http://www.mysql.com/.

[Sou] Sourceforge. *StrutsTestCase for JUnit v2.1.4*. Sourceforge, http://strutstestcase.sourceforge.net/.