Universität Zürich
Institut für Informatik

Dynamic and Distributed
Information Systems

Ein regel- und statistikbasiertes Empfehlungssystem für das Masterstudium in Informatik

Bachelorarbeit vom 3. Sept. 2007

Peter Höltschi
von Zürich, Schweiz

Matrikel-Nr.: 03-718-269
phoeltschi@access.uzh.ch

Betreuerin: **Esther Kaufmann**

Prof. Abraham Bernstein, PhD
Departement of Informatics
University of Zurich
http://www.ifi.uzh.ch/ddis

Zusammenfassung

In dieser Bachelorarbeit wird ein regel- und statistikbasiertes Empfehlungssystem für die Planung des Masterstudiums in Informatik an der Universität Zürich spezifiziert, entworfen und an einem Prototypen erprobt. Das System unterstützt Informatikstudierende bei der automatischen Erstellung von Studienplänen. Dadurch wird zum einen die Einhaltung der Studienreglemente garantiert. Andererseits erhalten die Studierenden ein Bild darüber, wie ihr Masterstudium aussehen könnte. Sie müssen dazu die Daten ihres Leistungsausweises zur Verfügung stellen und Präferenzen zur Studienrichtung und zur Modulwahl angeben. Aufgrund dieser Daten erstellt das System mittels mehrerer Filter- und Sortierfunktionen die gewünschten Studienpläne. In einer Evaluation wurden Studierende um die manuelle Erstellung eines Studienplans und der Angabe der Daten zur automatischen Erstellung angefragt. Eine Analyse der Resultate und ein Vergleich zwischen dem manuellen und dem automatisch erstellten Studienplan hat ergeben, dass die Qualität von letzterem stark von der Qualität und der Menge der Präferenzangaben des Studenten abhängt. Zudem kam heraus, dass das System zur optimalen Nutzung mit zusätzlichen Features ausgestattet werden sollte.

Abstract

This Bachelor Thesis describes the specification, design and implementation of a rule- and statistics based recommendation system for the planning of the master study in informatics at the University of Zurich. The system supports students in automatically generating study plans. On one hand, this guaranties the compliance with the reglements of study. On the other hand, the students quickly get a picture of how a master study plan can look like. For this to work, the student has to provide data of his transcript of records, some details concerning his course of study and a choice of preferred lecture contents. Based on this data, the system generates the study plans using filtering and sorting functions. In an evaluation, some students were asked to provide a manually created study plan and the data for automatically generating study plans. The analysis of the results and a comparison of the manually and automatically generated study plans showed that the quality and quantity of the provided data have a strong impact on the quality of the resulting study plans. To enhance the system, further features should be implemented.

Danksagung

Ich möchte meiner Betreuerin Esther Kaufmann für die hilfreiche und lehrreiche Beratung zu dieser Arbeit danken. Zudem danke ich Christoph Kiefer für die Idee den Rucksack-Algorithmus einzusetzen. Im Weiteren danke ich Corinne Maurer und Katrin Häsler für die Bereitstellung wichtiger Informationen zu administrativen Prozessen. Ebenfalls danke ich auch Roman Steinlin von der Redaktion VVZ für den Zugang ins Vorlesungsverzeichnissystem und meinen Mitstudenten für die Erstellung der nötigen Evaluationsdaten.

Zum Schluss geht ein Dank an alle die mich während meiner Studienzeit tatkräftig unterstützt und begleitet haben.

Inhaltsverzeichnis

1 Einleitung

Diese Arbeit befasst sich mit der Spezifikation, dem Entwurf und einer prototypischen Implementierung eines Empfehlungssystems für das Masterstudium in Informatik an der Universität Zürich. Dieses Empfehlungssystem generiert Studienpläne aufgrund von Präferenzangaben des Studenten. Der Lesbarkeit halber werden in dieser Arbeit für Personenbezeichnungen die männlichen Formen verwendet, wobei die weiblichen Formen stets mitgemeint sind.

1.1 Problemstellung

Das Masterstudium in Informatik an der Universität Zürich ist, im Gegensatz zum Bachelorstudium, relativ offen aufgebaut. Für die drei angebotenen Studienrichtungen werden nur wenige Pflichtveranstaltungen vorgeschrieben, sodass die Studenten sich ihren Studienplan selbständig, nach ihren Präferenzen aufbauen können.

Die Reglemente zum Masterstudium sind, da sie als rechtsverbindliche Regeln und nicht als Hilfestellungen für die Studienwahl geschrieben wurden, für die Studenten oft schwer verständlich. Zudem kann sich nicht jeder Interessierende vorstellen, was ihn im Masterstudium in Informatik erwarten könnte, respektive wie ein komplettes über mindestens 4 Semester laufendes Voll- oder Teilzeitstudium aussehen könnte.

Zu diesem Zweck wurde das Projekt MyMaster@IFI[1] ins Leben gerufen. Es soll den Studierenden eine Orientierungshilfe schaffen, wo sie sich selbständig mittels einer so genannten *Student Self Service*-Anwendung ihr individuelles Studium planen können. Zu diesem Projekt existieren bis heute nur eine grobe Projektbeschreibung und zahlreiche Ideen in den Köpfen einer kleinen Gruppe von Masterstudenten, die sich im Rahmen eines Kolloquiums geleitet durch Prof. A. Bernstein dazu einige Gedanken gemacht haben.

Ein Bestandteil dieses Projekts ist ein Empfehlungssystem. Dieses wird in vorliegender Arbeit entworfen und soll durch Einbezug mehrerer Datenquellen und durch Präferenzangaben des Studierenden mögliche Studienpläne generieren. Durch einen *regelbasierten Filter*[2] sollen die Reglemente in formal-logische Regeln umgewandelt werden, sodass eine maschinelle Verarbeitung möglich wird. Dadurch wird die Menge der möglichen, anrechenbaren Module ermittelt. Durch einen *statistikbasierten Filter*[3] sollen die präferierten Module ermittelt werden. Die resultierenden Module können dann zur Generierung von Studienplänen verwendet werden.

[1] Institut für Informatik
[2] Filterfunktion, die aufgrund von Regeln eine Menge von Daten filtriert.
[3] Filterfunktion, die mittels einer Suchmaschine relevante Daten filtriert.

1.2 Zielsetzung

Ziel dieser Arbeit ist es, ein regel- und statistikbasiertes Empfehlungssystem für das Masterstudium in Informatik im Sinne von Abschnitt 1.1 zu spezifizieren, zu entwerfen und prototypisch zu implementieren. Die Machbarkeit soll durch möglichst hohe Anforderungsabdeckung konzeptionell und am Prototyp analysiert werden. Die Evaluation des Systems hat den Zweck den Mehrwert (Gebrauchswert/Nutzen) der automatischen Erstellung eines Studienplans gegenüber der manuellen darzulegen. Dazu werden drei angehende Masterstudenten der Informatik um eine manuelle Studienplanung und einiger Zusatzdaten angefragt. Nach Erhalt der Evaluationsdaten sind die Studienpläne automatisch durch das Empfehlungssystem anzufertigen. Eine Auswertung beider Resultate zeigt dann den Mehrwert dieses Systems auf. Beschränkungen und künftige Arbeiten werden ebenfalls dargelegt.

1.3 Gliederung

In Kapitel 2 werden im Wesentlichen das Gegebene aus dem Systemkontext und die Anforderungen an das Empfehlungssystem spezifiziert. Kapitel 3 befasst sich mit dem Entwurf des Systems. Es wird der Datenverarbeitungsprozess des Systems vorgestellt und Designentscheidungen zu den einzelnen Bestandteilen erläutert. In Kapitel 4 wird eine Evaluation zum Mehrwert des Systems durchgeführt, und in Kapitel 5 und 6 werden Beschränkungen sowie künftige Arbeiten aufgezeigt. Im abschliessenden Kapitel 7 werden Schlussfolgerungen dargelegt.

2 Spezifikation

In diesem Kapitel werden die zur Erstellung des Entwurfs nötigen Anforderungen spezifiziert. Diese Anforderungen wurden der Aufgabenstellung zur Bachelorarbeit und der Projektbeschreibung zum MyMaster@IFI-Projekt entnommen. Zusätzlich wurden die während der Ausarbeitung dieser Arbeit und die im Gespräch mit der Betreuerin Esther Kaufmann entdeckten nötigen Anforderungen aufgeführt. Eine detaillierte Bedürfnis-/Anforderungsanalyse mit Einbezug aller Anspruchsgruppen (*stakeholder)* dieses Systems wurde aufgrund der Rahmenbedingungen dieser Arbeit nicht durchgeführt.

In Abschnitt 2.1 wird der Systemkontext beschrieben. In Abschnitt 2.2 werden aus den Reglementen die formal-logischen Regeln extrahiert und im darauf folgenden Abschnitt 2.3 werden die Eingabedaten spezifiziert. Der Aufbau des Masterstudiums in Informatik wird in Abschnitt 2.4 schematisch dargestellt.

2.1 Systemkontext

Das in Abbildung 1 dargestellte Kontextdiagramm zeigt die Einbettung des Prototyps in den Systemkontext. Ein Benutzer resp. der Studierende greift via Benutzerschnittstelle auf das Empfehlungssystem zu. Dazu werden vom System zum einen der *Leistungsausweis*[4] und zum anderen eine Reihe von Präferenzangaben zum Studium verlangt. Aufgrund von sicherheits- und datenschutzrelevanten Aspekten wurde ein *Direktzugriff*[5] auf das *Modulbuchungssystem*[6], wovon die Leistungsnachweise abgerufen werden können, nicht zugelassen. Als Alternative werden die Daten aus der Web-Benutzerschnittstelle des Modulbuchungssystems in Textform kopiert und an das Empfehlungssystem übertragen. Das Empfehlungssystem greift daraufhin auf das *Vorlesungsverzeichnissystem*[7] zu, um die nötigen Daten zur Generierung der Studienpläne zu erhalten. Für den Zugriff auf das Vorlesungsverzeichnissystem gelten dieselben Auflagen wie beim Modulbuchungssystem. Um bei der Erstellung des Prototyps dennoch mit realen Daten arbeiten zu können, wurde ein gesicherter Web-Zugriff auf Vorlesungsinhalte in Dateiform gewährt. Zur Laufzeit greift der in dieser Arbeit erstellte Prototyp daher nicht direkt auf das Vorlesungsverzeichnissystem zu, sondern auf die lokal heruntergeladenen Dateien. Es gibt somit im heutigen Setup keinen Direktzugriff auf die umliegenden Systeme. Der dargestellte Zugriff auf die Reglemente ist implizit zu verstehen. In Wirklichkeit werden die Regeln aus den Reglementen ins Empfehlungssystem übersetzt übertragen.

[4] Liste der gebuchten, erfolgreich und allenfalls nicht erfolgreich abgeschlossenen Module
[5] Zugriff (via API oder Abfragesprache) auf die Datenbank des Systems
[6] System zur Buchung von Modulen und zum Abrufen des Leistungsnachweises. Unter einem (Studien-)Modul werden alle Informationen zu einer Veranstaltung (Vorlesung, Seminar, ...) gekapselt.
[7] Öffentlich zugängliches Verzeichnis aller Veranstaltungen (Studienmodule) an der Universität Zürich

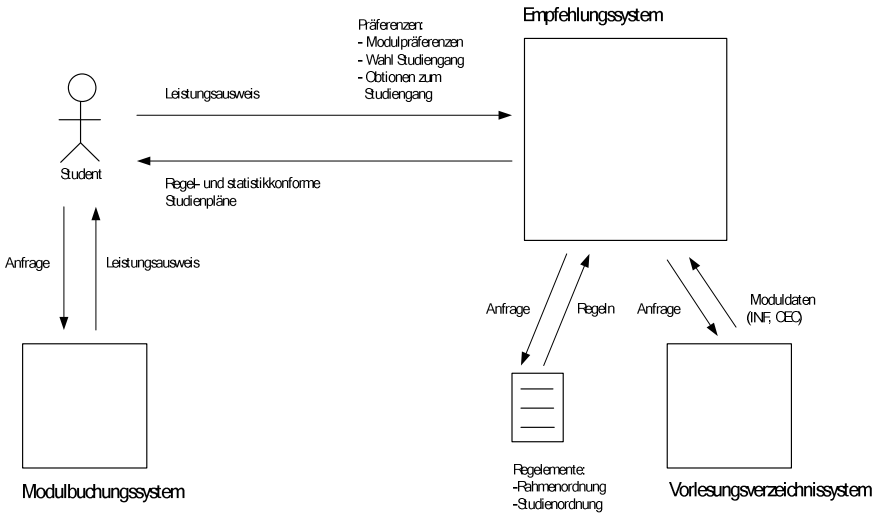

Abbildung 1: Kontextdiagramm zum Empfehlungssystem

2.2 Spezifikation der Regelmenge

Die in Prosa verfassten Reglemente des Masterstudiums in Informatik müssen zur maschinellen Verwendung in einem Regelsystem formal-logisch umgewandelt werden. Es werden dabei nur die für das Empfehlungssystem relevanten Regeln übernommen. Regeln, die eine Abklärung durch das *Lehrbereichssekretariat*[8] voraussetzen, sind nicht relevant.

2.2.1 Reglemente des Masterstudiums

Für das Masterstudium in Informatik stehen dem Studenten zwei Reglemente zur Verfügung:

- Rahmenordnung
- Studienordnung

Eine Prüfung der beiden Reglemente hat ergeben, dass für die Spezifikation der Regeln ausschliesslich die *Studienordnung* relevant ist. Relevante Bereiche der Rahmenordnung sind in der Studienordnung ebenfalls aufgeführt.

2.2.2 Extrahierte Regeln

Eine Regel im formalen Sinne besitzt eine *Prämisse* (Voraussetzung) und eine *Konklusion* (Folgerung). Die Prämisse bildet das WENN (*if*) und die Konklusion das DANN (*then*) einer Regel. In Tabelle 1 ist eine Teilmenge der extrahierten Regeln aus der Studienordnung aufgeführt. Alle weiteren formalisierbaren Bereiche der Studienordnung können Anhang B entnommen werden. Zur Beschreibung der Regeln wurde eine semiformale Syntax gewählt, damit auch Nicht-Informatiker diese inhaltlich prüfen können. Die fett markierten Regelbestandteile sind diejenigen, die im regelbasierten Filter in Abschnitt 3.3.2 eingebunden wurden. Sie werden in eine für den regelbasierten Filter verständliche Syntax überführt. Kursiv markiert bedeutet, dass die Regel im Empfehlungssystem zur Anwendung kommt, jedoch nicht Teil des regelbasierten

[8] Administration und Anlaufstelle für Informatikstudenten

Filters ist. Die restlichen Regeln setzten (zurzeit noch) manuelle Handlungen durch das Lehrbereichssekretariat voraus und sind somit nicht auf das Empfehlungssystem anwendbar.

Regelmengen können auch durch so genannte *Aktionspläne (action plans)* dargestellt *und verifiziert werden*. In der Computerwissenschaft befasst sich der Bereich *Planning* damit. Mittels einer formalen Sprache können solche Aktionspläne auf ihre Korrektheit hin geprüft werden, wobei eine Regel in die Form *Aktion-Vorbedingung-Nachbedingung* (*action-precondition-effect*) gebracht wird. Ist die Vorbedingung erfüllt, wird die Aktion ausgeführt, und es resultiert eine Nachbedingung (vergleiche [Russell & Norvig 2003]). Für den operativen Einsatz wird empfohlen die Regelmenge bei jeder Änderung einer Regel mittels eines *Unit-Tests*[9] zu verifizieren.

Zu Beginn besitzen alle Module den Status *buchbar*. Durch folgende Regeln können sie den Status *nicht buchbar* und *präferiert* erhalten. Die Regel *Check_successefully_passed_modules* überprüft, ob das aktuelle Modul mit einem aus dem Leistungsausweis übereinstimmt. Zudem wird geschaut, ob das passende Modul im Leistungsausweis den Buchungsstatus *erfolgreich abgeschlossen* besitzt. Ist dem so, kann das Modul nicht noch einmal gebucht werden. Die Regel *Check_accountability* überprüft ob ein bereits gebuchtes Modul auf Masternivau existiert, dass aufgrund der vorherigen Regel den Status *nicht buchbar* erhalten hat. Ist dies der Fall, werden die minimal zu erreichenden *Kreditpunkte (ECTS)* um jene dieses Moduls dekrementiert. Diese Regel kann zu einem falschen Ergebnis führen. Hat ein Student ein Modul, dass sowohl für den Master- als auch für den Bachelorabschluss angerechnet werden kann, bereits für den Bachelorabschluss angerechnet, ist eine Anrechnung für den Masterabschluss nicht mehr möglich. Ob ein Modul bereits für den Bachelorabschluss angerechnet wurde, geht jedoch nicht aus dem (elektronischen) Leistungsausweis hervor (aktueller Kenntnisstand). Eine Lösung für dieses Problem wäre, dass der Student selber angibt, welche Module er bereits für den Bachelorabschluss angerechnet hat resp. welche er für den Masterabschluss anrechnen möchte. Dazu wurde der Regelbestandteil *anrechenbar* hinzugefügt. Die nächsten beiden Regeln, *Check_master_niveau* und *Check_precondition* überprüfen, ob ein Modul auf Masterniveau liegt und ob die Modulvoraussetzungen erfüllt sind, daher ob man die vorausgesetzten Module bereits besucht und erfolgreich abgeschlossen hat. Die letzte Regel *Check_course_of_study_for_required_modules* überprüft, ob ein Modul ein Pflichtmodul des Masterstudiums in Informatik ist, und wenn ja, ob es zur gewählten *Studienrichtung*[10] passt.

[9] Verfahren zur Verifikation von Softwaremodulen

[10] Mögliche Studienrichtungen: Information Systems, Software Systems, Cognitive Systems

Abschnitt*	Regelname	when	then
4.3, 6.1	Check_successefully_ passed_modules	**Modul erfolgreich abgeschlossen** OR Modul inhaltlich gleichartig oder ähnlich zu einem erfolgreich abgeschlossenen Modul OR Neuanrechnung des Moduls aus zeitlichen Gründen	**Modul nicht buchbar**
2.5, 7.1	Check_accountability	**Modul auf Masterniveau AND nicht buchbar AND anrechenbar**	**Minimal zu erwerbende Anzahl ECTS -= ECTS vom Modul**
5.2	Check_master_niveau	**Modul NOT auf Masterniveau**	**Modul nicht buchbar**
5.1	Check_course_of_study_ for_required_modules	**(Modul = Master-Basismodul OR Modul = Projektarbeit OR Modul = Masterarbeit) AND NOT Modul gehört zur Studienrichtung**	**Modul nicht buchbar**
4.1	Check_precondition	**Modul-Voraussetzung (en) erfüllt**	**Modul buchbar**

* Studienordnung Version 1.1 vom 13. Juni 2007

Tabelle 1: Extrahierte Regeln aus der Studienordnung

2.3 Spezifikation der Eingabedaten

Zu den Eingabedaten zählen sowohl die vom Benutzer eingegebenen Präferenzangaben zum Studium als auch Daten zu den Modulen aus dem Modulbuchungssystem und dem Vorlesungsverzeichnissystem. Nachfolgend wird erläutert, welche Bestandteile dieser Datenbestände für die Verarbeitung im Empfehlungssystem relevant sind und welchen Gesetzen (SOLL-Anforderungen) diese Daten folgen müssen, damit eine maschinelle Verarbeitung mit vernünftigem Aufwand möglich ist. Dies ist nötig, da der IST-Zustand nicht in jedem Fall diesen Anforderungen entspricht.

2.3.1 Daten aus dem Vorlesungsverzeichnissystem

Wie bereits angesprochen, können die Moduldaten aus dem Vorlesungsverzeichnissystem nicht mittels eines Direktzugriffs abgerufen werden. Über einen speziellen Zugang hat ein autorisierter Benutzer jedoch Zugriff auf aktuelle Vorlesungsdaten, die in Dateiform vorliegen. Es handelt sich um Datenbankexporte, die als Trennzeichen ein Doppelhochkomma besitzen.

Sie werden fortan *VVZ-Exporte* genannt. Dieser Zugang wurde von der Redaktion VVZ, der Verwaltungsstelle des Vorlesungsverzeichnisses, zeitlimitiert zur Verfügung gestellt. Über diesen Zugang hat man zwei Optionen, um die Vorlesungsdaten in Form von Dateien herunterzuladen:

- Export nach Studiengang (Studienrichtung)
- Export nach Anbieter

Unter *Export nach Anbieter* kann man sich alle Module einer Fakultät in einer Datei herunterladen. Für diese Arbeit wurde jedoch die Option *Export nach Studiengang* (*Studienrichtung*) verwendet, da nur über diesen Weg die Module auf die einzelnen Studienrichtungen aufgeteilt werden können. Dies ist nötig, weil die Datensätze der VVZ-Exporte keine Informationen zur Studienrichtung enthalten. In Abschnitt 3.3.1 wird beschrieben, wie diese Dateien *geparst*[11] werden.

Ein Modul im Vorlesungsverzeichnissystem besitzt mehrere Attribute. Die für die Verarbeitung mit dem Empfehlungssystem wichtigen SOLL-Attribute sind nachfolgend aufgelistet:

- a Modulkürzel
- b Veranstaltungstyp (z.B. Vorlesung, Seminar, Praxisorientierte Veranst.)
- c Genre (z.B. Informatik oder Wirtschaftswissenschaften)
- d Studienrichtung (z.B. Software Systems, Information Systems)
- e Titel
- f Voraussetzung
- g Beschreibung
- h ECTS-Punkte
- i Niveau (Assessement-, Bachelor- oder Masterstufe)
- j Semestertyp (Herbstsemester (HS) oder Frühlingssemester (FS))

Diese Informationen sind zwingend notwendig, damit eine komplette Regelprüfung durchgeführt werden kann und entschieden werden kann, ob ein Modul in einen Studienplan aufgenommen werden soll oder nicht.

Das Modulkürzel wird nachfolgend mittels eines endlichen Automaten (*deterministic finite state automaton*) spezifiziert. Dies ist nötig, damit das Kürzel, wie in Abschnitt 3.3.1 beschrieben, korrekt geparst werden kann. *03SMMINF4502* ist ein Beispiel für ein solches Modulkürzel. Es stammt von der Veranstaltung „Requirements Engineering II“ bei Prof. M. Glinz. Das Modulkürzel ist aus folgenden Bestandteilen aufgebaut:

- Fakultät: 03 (Wirtschaftswissenschaftliche Fakultät)
- Modultyp: SM (Studien-Modul)
- Niveau: A,B oder M
- Genre: INF, OEC oder weitere
- Veranstaltungstyp: P, S, (ohne entspricht einer Vorlesung VL)
- Veranstaltungsnummer: 4-stellige Zahl

[11] Analyse der Eingabedaten und Umwandlung in ein gewünschtes Format zur Weiterverarbeitung

Ein endlicher Automat für das Modulkürzel kann Abbildung 2 entnommen werden. Dieser Automat zeigt die Aufbaumöglichkeiten eines gültigen Modulkürzels an der Wirtschaftswissenschaftlichen Fakultät. Neben den aufgeführten Niveaukürzeln werden noch andere eingesetzt. Diese sind für den (vor den Masterstudiengängen angebotenen) *Diplomstudiengang* oder *Doktorandenseminare* bestimmt. In dieser Arbeit wird jedoch nur die Studienordnung des Masterstudiums in Informatik betrachtet. Es kommt vor, dass ein Modulkürzel kein Niveaukürzel besitzt. Dann folgt auf das *Präfix* (*Fakultät und Modultyp*) direkt das Genrekürzel. Wenn der Veranstaltungstyp eine Vorlesung (VL) ist, dann wird kein Kürzel geschrieben. Endet man bei einer Prüfung im Zustand i, handelt es sich um ein gültiges Modulkürzel.

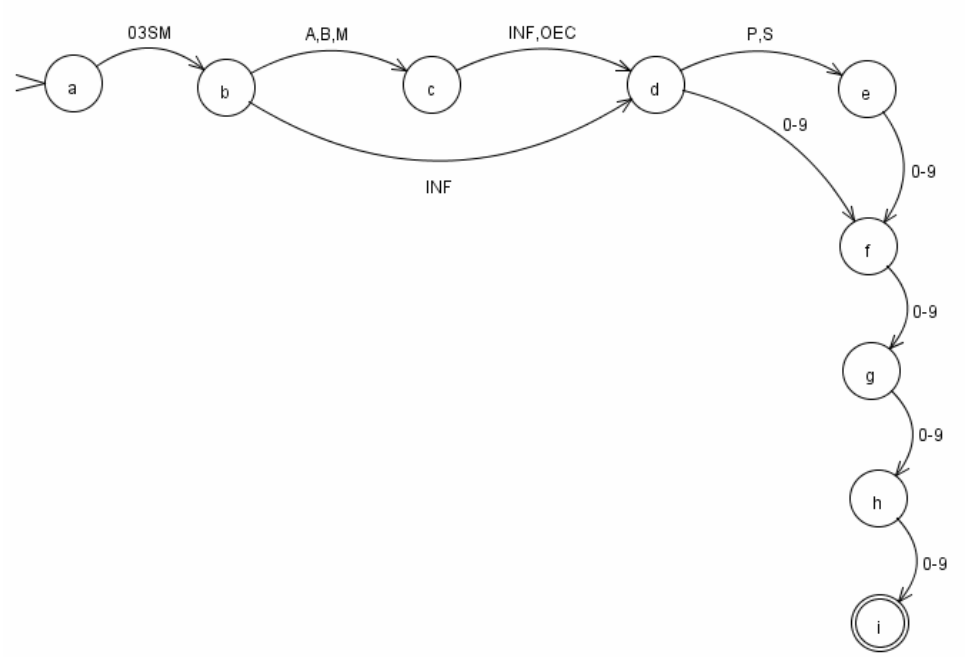

Abbildung 2: Spezifikation der Modulkürzel durch einen endlichen Automaten

Durch das Parsen des Modulkürzels können Attributwerte der Attribute *a*, *b*, *c*, und *i* ermittelt werden. Wenn der Parser innerhalb eines VVZ-Exports ein Modulkürzel antrifft und dieses erfolgreich geparst hat, können die weiteren Attributwerte, die durch ein Trennzeichen getrennt sind, immer in derselben Reihenfolge ausgelesen werden.

Einzige Ausnahme bildet Attribut *f* (Voraussetzung). SOLL-Anforderung an Attribut f wäre, dass dort eine Referenz zu einem oder mehreren Modulen stehen würde. Voraussetzung für obige Beispielvorlesung ist beispielsweise die Vorlesung „Requirements Engineering I“ mit dem Modulkürzel *03SMMINF4204*. Stünde unter Voraussetzung nur dieses Modulkürzel wären die SOLL-Anforderungen erfüllt. Die Voraussetzungen eines Moduls werden bei der digitalen Erfassung jedoch in ein Freitextfeld eingegeben, und es existieren keine formalen Einschränkungen bei der Dateneingabe. Dies wurde mit der zuständigen Person am Institut für Informatik (IFI) abgeklärt. Das Parsen der Modulvoraussetzungen ist daher keine triviale Aufgabe. Es ist jedoch anzumerken, dass es zumindest bei den Informatikmodulen zurzeit nur wenige Module gibt, die andere Module auf Masterniveau voraussetzen.

2.3.2 Daten aus dem Modulbuchungssystem

Im Modulbuchungssystem kann der Student über eine Web-Benutzerschnittstelle seinen Leistungsausweis einsehen. Darin ist vermerkt, welche Module er gebucht hat und welchen Status die gebuchten Module besitzen. Folgende Daten werden fürs Empfehlungssystem benötigt:

- Modulkürzel
- Buchungsstatus, z.B. „Erfolgreich abgeschlossen“, „gebucht“, usw.

Wird eine Kopie dieser Daten ans Empfehlungssystem übertragen, werden die Daten geparst. Im regelbasierten Filter werden sie für die Prüfung mit der Regel *Check_successefully_passed_modules* benötigt.

2.3.3 Optionen zur Studienwahl

Bei der Erstellung der Studienpläne stehen (neben den Präferenzangaben zur Modulwahl) folgende Parameter dem Benutzer zur Auswahl:

- a Studienrichtung (Software Systems, Information Systems, usw.)
- b Zeiteinteilung (Voll- oder Teilzeitstudium)
- c Nebenfach (Ja/Nein)
- (d Sprache, z.B. Deutsch oder Englisch)

Die Option d steht zurzeit noch nicht zur Auswahl, ist aber denkbar für zukünftige Jahrgänge.

2.3.4 Freitext-Präferenzen zur Modulwahl

Damit ein Student Studienpläne erhält, die optimal seinen Bedürfnissen entsprechen, kann er seine Präferenzen angeben, die bei der Wahl der Module berücksichtig werden. Diese können in Form von (zusammengesetzten) Begriffen entweder frei eingegeben werden und/oder, wie in Abschnitt 2.3.5 beschrieben, aus einer Menge von Kategorien ausgewählt werden. Eine Eingabe könnte folgendermassen aussehen:

- Anforderungsspezifikation, Model Driven Developement, Service Oriented Computing, Projektmanagement

Die Module werden aufgrund dieser Suchbegriffe durchsucht. Konkret werden Attribut *e* und *g* (Titel und Beschreibung) eines Moduls durchsucht. Dies ist Teil der statistikbasierten Filterung und kann Abschnitt 3.3.3 entnommen werden. Das Resultat dieses Filtervorgangs sind die präferierten Module.

2.3.5 Kategoriale Präferenzen zur Modulwahl

Analog zu Abschnitt 2.3.4 ist es auch möglich, aus einer *Taxonomie*[12] eine oder mehrere Kategorien als Präferenz(en) auszuwählen. Als Referenztaxonomie wurde das *ACM-Kategoriensystem* gewählt. Dieser Standard kommt in Bibliotheken zum Einsatz und wird benutzt, um Literatur im Bereich *Computer Science* einheitlich zu ordnen. Die ACM-Kategorien können unter

[12] Im klassischen Sinne: Menge von Kategorien, die hierarchisch angeordnet sind

http://www.jucs.org/jucs_info/acm_categories

eingesehen werden. Zur Verwendung mit dem Empfehlungssystem wurde eine geeignete Untermenge davon ausgewählt, die möglichst gut zum heutigen Angebot an Veranstaltungen am Institut für Informatik passt. Diese Liste kann Anhang A entnommen werden.

2.4 Aufbau des Masterstudiums

Die Studienordnung für den Master in Informatik enthält eine Reihe von Richtlinien, welche die Modulwahl einschränken. Diese Einschränkungen sind in Abbildung 3 graphisch dargestellt. In den 4 orange hinterlegten *Modulgruppen* (*Informatik Seminar*, *Informatik Wahlpflichtmodule*, *Informatik Wahlmodule* und *Freie Wahlmodule*) muss mindestens die in Klammern angegebene Anzahl ECTS-Punkte erreicht werden. Der Studienordnung kann auch noch ein Maximum anrechenbarer Module entnommen werden. Die Module dieser Modulgruppen können beliebig über die Semester verteilt besucht werden. Die Darstellung hat daher nur schematischen Charakter.

Die blau hinterlegten *Pflichtmodule* können hingegen nicht in einem beliebigen Semester besucht werden. Abbildung 4 zeigt die Einschränkungen. In dieser Abbildung wird von einem Vollzeitstudium mit Beginn in einem Herbstsemester ausgegangen. Das *Master-Basis Modul* kann beispielsweise nur resp. muss entweder im ersten oder im zweiten Semester des Masterstudiums belegt werden. Diese Einschränkungen begrenzen die Anzahl möglicher Studienpläne.

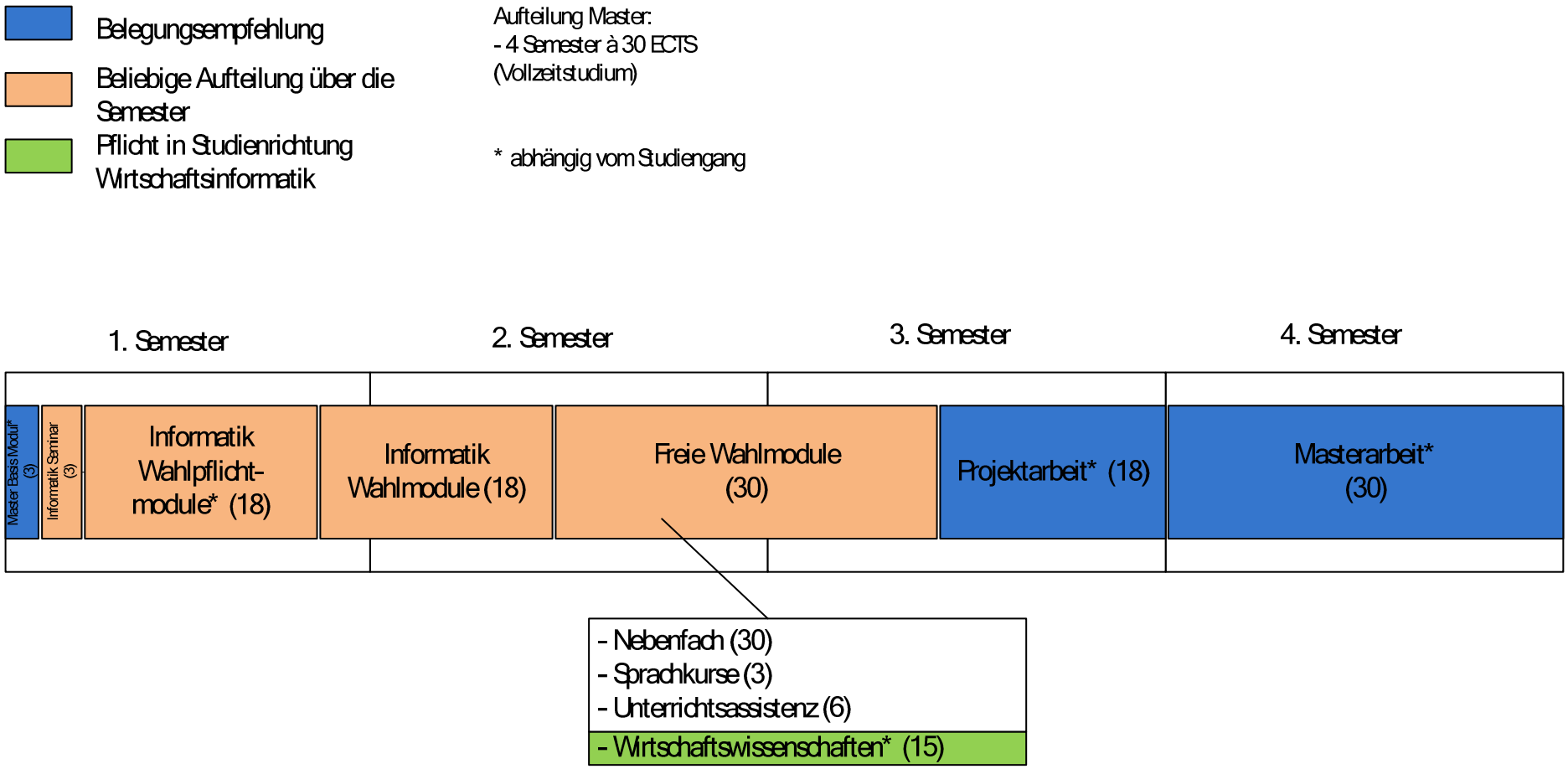

Abbildung 3: Aufbau des Masterstudiums in Informatik

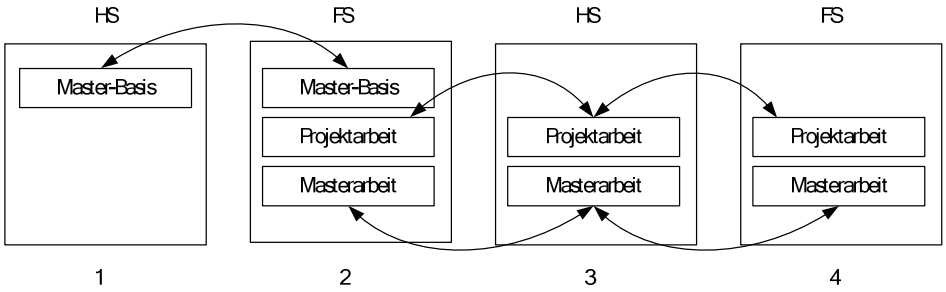

Abbildung 4: Belegungsmöglichkeiten der Pflichtmodule im Masterstudium

3 Entwurf und Implementierung

In diesem Kapitel werden Details zum Entwurf und der Implementierung des Demonstrationsprototypen beschrieben und, wo nötig, anhand von graphischen Modellen präzisiert. Dazu werden die in Kapitel 2 spezifizierten Anforderungen als Grundlage verwendet. Der Quellcode des Prototyps, dessen Dokumentation und weitere Dateien sind auf Medium 1 (siehe Anhang D) gespeichert.

3.1 Prozess der Studienplan-Generierung

Die Studienplan-Generierung ist der zentrale Prozess des Empfehlungssystems und ist in Abbildung 5 dargestellt.

Abbildung 5: EPK der Modul-Datenverarbeitung

Die ersten beiden Prozessfunktionen stösst der Benutzer an. Sie verarbeiten die Eingaben des Benutzers. Eine Skizze der Benutzerschnittstelle ist in Abschnitt

3.2 einsehbar. Die darauf folgenden beiden Parsing-Funktionen bilden die Daten-Vorverarbeitung (*data preprocessing*) der Moduldaten und des Leistungsausweises (Buchungsdaten). Sie sind Input für die *regelbasierte Filterung*. Bei dieser Funktion werden die Module aufgrund der Regeln geprüft und gefiltert. Das Ergebnis dieses Prozessschrittes wird der *statistikbasierten Filterung* übergeben. Es resultieren *mögliche* inkl. *präferierte Module*. Sie werden dem *Studienplan-Generator* übergeben. Dort findet in einem ersten Schritt eine Aufteilung der Module in die Modulgruppen statt. Diese werden daraufhin über die einzelnen Semester verteilt. Zuletzt folgt die Ausgabe der Studienpläne. Bis zur Prozessfunktion *regelbasierte Filterung* ist es möglich, die Prozesskette in unterschiedlicher Reihenfolge ablaufen zu lassen. Es sind daher mehrere *Partial-Order-Plans*[13] möglich, die zu *Total-Order-Plans*[14] *linearisiert*[15] werden können (vergleiche [Russell & Norvig 2003]). Ab der Prozessfunktion regelbasierte Filterung ist eine strickte Einhaltung der Reihenfolge notwendig.

3.2 Skizze der Benutzerschnittstelle

In Abbildung 6 wird eine Skizze einer möglichen Benutzerschnittstelle (Eingabemaske) des Empfehlungssystems gezeigt.

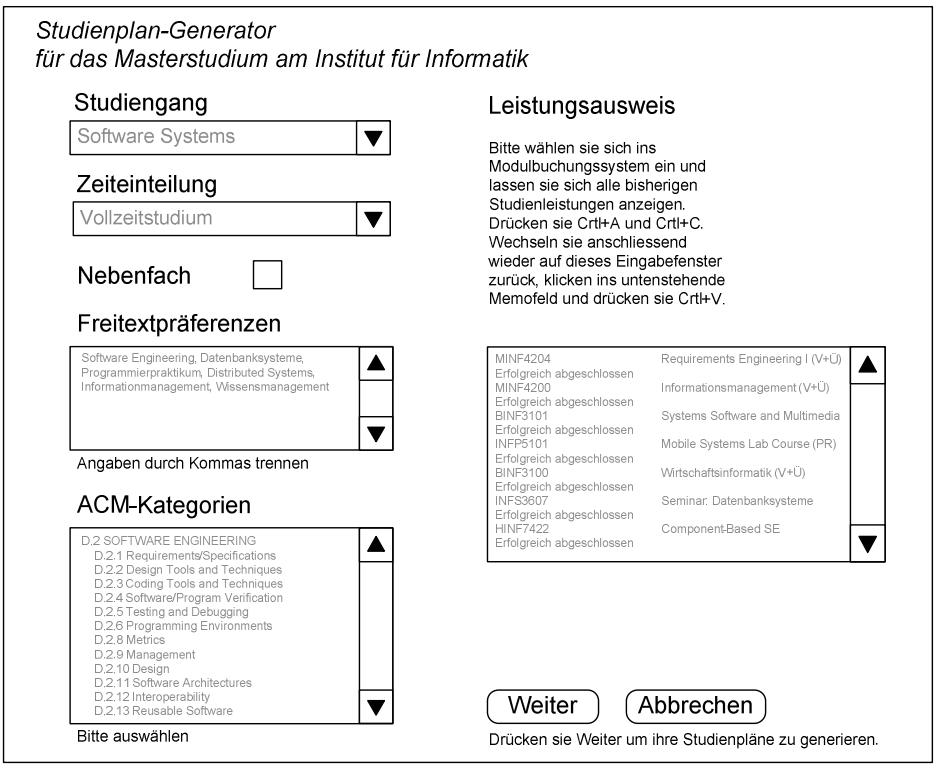

Abbildung 6: Skizze einer Eingabemaske des Empfehlungssystems

Zu allen Feldern muss eine (gültige) Eingabe gemacht werden. Grau hinterlegt sind Beispieleingaben zu sehen. Nach drücken der *Weiter*-Taste würden die generierten Studienpläne erscheinen, ähnlich dem Beispiel in Abschnitt 3.3.4 .

[13] Spezifizierung aller möglichen Ablaufvarianten. Kann durch einen Graphen dargestellt werden.
[14] Ein aus Partial-Order-Plans bestehender möglicher Gesamtablaufplan.
[15] In diesem Zusammenhang: Aneinanderreihung von Partial-Order-Plans.

3.3 Systemarchitektur

Das Empfehlungssystem besitzt eine *Typed Pipes & Filter*-Architektur. Typisiert deswegen, weil ausschliesslich Daten vom Typ *Module* verarbeitet werden können/müssen. Die Ausgabe (*output*), die Ergebnismenge, des einen Filters ist die Eingabe (*input*) des nächsten. Nachfolgend wird auf Charakteristika, Vor- und Nachteile dieser Architektur eingegangen.

Im klassischen Sinne werden die Pipes & Filter ineinander verschachtelt. Bei diesem System sind die Filter jedoch seriell aneinander gereiht. Zudem können die Filter nicht beliebig aneinandergereiht werden. Dies wurde bereits in Abschnitt 3.1 festgehalten. Es ist jedoch möglich, die Implementierung der jeweiligen Filter beliebig auszutauschen, solange die Schnittstellenverträge (*design by contract*) eingehalten werden. Dies erhöht die Wartbarkeit (*maintainabillity*) des Systems. Das im Voraus stattfindende einmalige Parsen der Daten reduziert den Aufwand jedes einzelnen Filters, da ansonsten jeder einzelne dies tun müsste. Über Herausforderungen hinsichtlich Nebenläufigkeit braucht man sich bei dieser Art der Architektur keine Sorgen zu machen, da der Datenstrom an Moduldaten die einzelnen Filter synchronisiert. Dieser Vorteil ist auch Nachteil, falls Änderungen an den Ausgangsdaten vorgenommen werden, da dies wahrscheinlich auch zu Anpassungen der Implementierung führt (vergleiche [Glinz & Gall 2005]). Die Dokumentation der einzelnen Bausteine des Systems sollte jedoch genügend Transparenz bieten, um mit wenig Aufwand Änderungen dieser Art vorzunehmen. Die einzelnen Bausteine des Empfehlungssystems und deren gegenseitige Interaktionen sind im Sequenzdiagramm der Abbildung 7 dargestellt. Die Klasse *Main* ist das Verbindungsglied zwischen den weiteren Klassen (Objekten). Sie stösst nach Erhalt der Benutzerdaten (*Study*-Objekt und *Preferenc*e-Objekte) und des Leistungsnachweises den *TextParser* an und liefert jeweils einen Vektor vom Typ Module zurück. Alle nötigen Daten werden der *RuleEngine* übergeben. Nach deren Start durch die Methode *fireFules()* wird jedes Modul gefiltert und entsprechend markiert. Danach wird noch die *SearchEngine* benutzt, um die statistikbasierte Filterung durchzuführen. Zuletzt folgt die Generierung der Studienpläne. Abbildung 7 und das nachfolgende Klassendiagramm aus Abbildung 8 dienen als Übersicht für die nachfolgenden Abschnitte.

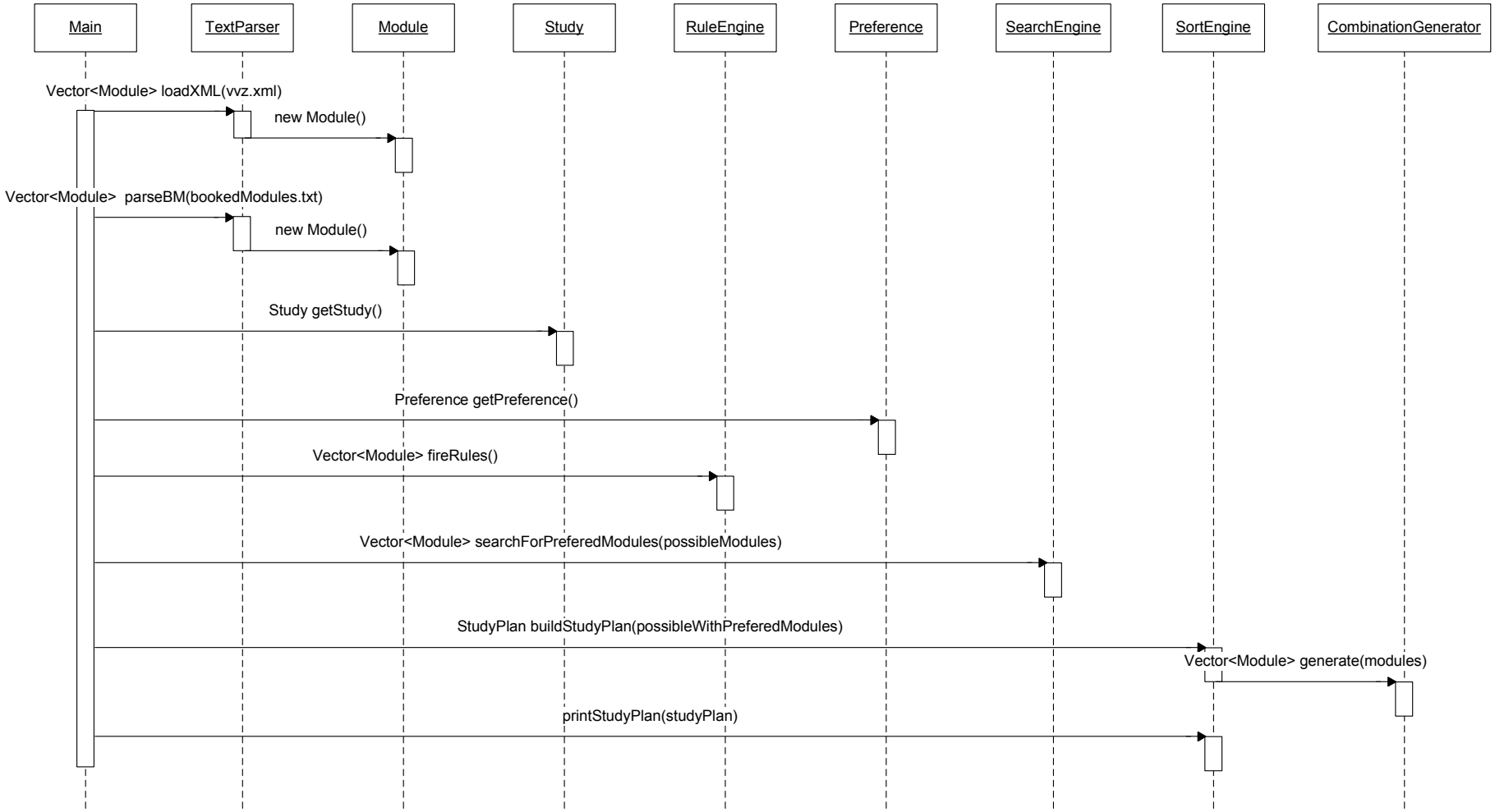

Abbildung 7: UML-Sequenzdiagramm des Empfehlungssystems

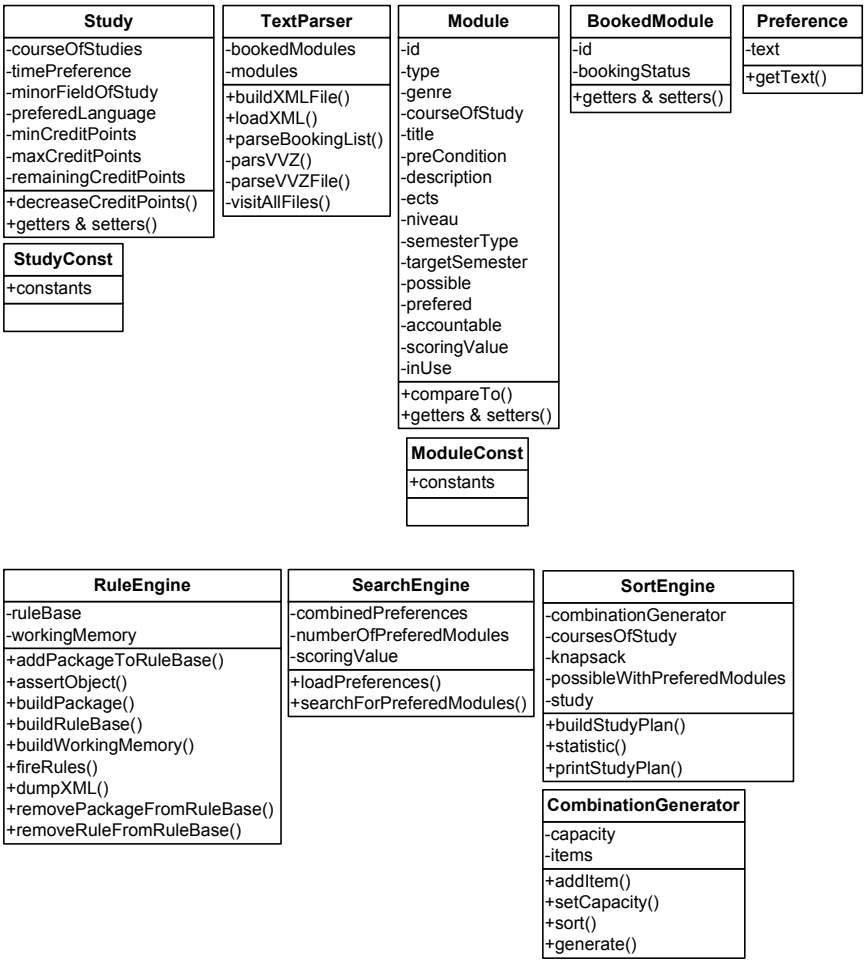

Abbildung 8: Klassendiagramm des Empfehlungssystems

3.3.1 Textparser

Der Textparser parst die VVZ-Exporte und den Leistungsausweis. Die Moduldaten aus den VVZ-Exporten werden in einem ersten Schritt in eine *XML-Repräsentation* transformiert (*Methode buildXMLFile()*). Dies geschieht ausserhalb der Laufzeit des Empfehlungssystems, da dieser Schritt je nach Grösse der zu transformierenden Datenmenge relativ lange dauert. Zur Laufzeit des Systems werden die *XML-Dateien* geladen (*Methode loadXML()*). Für den Prototyp werden zur Demonstration zwei XML-Dateien erstellt: *inf-modules.xml* und *oec-modules.xml*. In der erstgenannten Datei sind alle Informatikmodule und in der zweiten alle Wirtschaftswissenschaftsmodule abgespeichert. Von der Gesamtmenge aller zur Verfügung stehenden Daten werden nur jene, die in Abschnitt 2.3 spezifiziert wurden, benötigt und geparsed. Die Zuordnung der Informatikmodule zur passenden Studienrichtung konnte nicht automatisch durchgeführt werden, da zu diesem Zweck innerhalb eines Moduldatensatzes keine Angaben zur Verfügung stehen. Jedoch ist es möglich, wie bereits in Abschnitt 2.3.1 beschrieben, die Daten via Export nach Studiengang (Studienrichtung) herunterzuladen. Das Resultat sind 6 Textdateien, die nach dem Schema [Studienrichtungkürzel]-[Semestertypkürzel].txt abgespeichert werden:

- MMCS-HS.txt, MMCS-FS.txt, SOSY-HS.txt, SOSY-FS.txt, WI-HS.txt, WI-FS-txt

Der Parser berücksichtigt diese Informationen aus dem Dateinamen. Abbildung 9 zeigt einen Ausschnitt der Moduldaten in XML-Repräsentation.

```
<?xml version="1.0" encoding="UTF-8"?>
<Modules>
<Module id="MINF4211" title="Distributed Systems (V+Ü)"
        course_of_study="1" ects="3.0" genre="11"
        niveau="7" semester_type="13" type="8">
        <Description>
                Distributed computer systems (such as the Internet)
                haveradically changed our lives: they manage our
                infrastructures such astraffic control systems and/or stock
                exchanges, appear everywhere aroundus - be it in our cars or
                in our homes, and are, oftentimes, even thebasis for our
                entertainment. In most of theses applications
                multiplecomputers or systems work together. But how are
                these systemscommunicating, cooperating, and being
                coordinated to achieve theirgoals?
                …
        </Description>
        <Precondition>
                none
        </Precondition>
</Module>
        …
</Modules>
```

Abbildung 9: Ausschnitt der Moduldaten in XML-Repräsentation

Die Attributwerte aus Abbildung 9 können eigens definierte oder übernommene Werte sein. Nachfolgend die für den Prototyp verwendeten Attributwerte:

- course_of_study:
  - 0 Software Systems
  - 1 Information Systems
  - 2 Cognitive Systems
- genre:
  - 0 Informatikmodule
  - 1 Wirtschaftswissenschaftsmodule
- niveau:
  - 0 Assessementstufe
  - 1 Bachelorstufe
  - 2 Masterstufe
- semester_type:
  - 2 Herbstsemester
  - 3 Frühlingssemester
- type:
  - 0 Vorlesung (VL)
  - 1 Praxisorientierte Veranstaltung (PR)
  - 2 Seminar (SE)

Die Werte des Attributs *semester_type* sind vom Vorlesungsverzeichnissystem vorgegeben.

Zum Parsen der Moduldaten wird ein einfacher *String-Tokenizer*[16] verwendet, da die Daten bis auf das Modulkürzel wenig aufwändig sind zu parsen. Ein mächtigerer Parser wie ein *ANTLR*[17]*-Parser* wird daher nicht benötigt, könnte aber aus Gründen der Standardisierung und Wartbarkeit eingesetzt werden. Zur Umwandlung in die XML-Repräsentation kommt ein *DOM*[18]*-Parser* zum Einsatz. *JDOM* (Anhang E, Software 1) ist dazu geeignet und weit verbreitet. Die Daten aus dem Modulbuchungssystem werden analog zu denjenigen aus dem Vorlesungsverzeichnissystem geparst. Dies geschieht , wie bereits erwähnt, zur Laufzeit des Empfehlungssystems. Dadurch entfällt eine Überführung in die XML-Repräsentation.

3.3.2 Regelbasierter Filter

Liegen sowohl die Benutzer- als auch die Modul- und Buchungsdaten in Form von Objekten vor, können sie dem regelbasierten Filter übergeben werden. Der regelbasierte Filter ist ein *Expertensystem*, welches den Standard *Drools*[19] umsetzt. Die Wahl traf auf *Jboss Rules* (Anhang E, Software 2). Ein auf Drools basiertes Expertensystem wurde gegenüber beispielsweise *Jess*[20] bevorzugt, weil die Syntax von Drools ähnlich den Sprachen ist, die Informatiker an der UZH kennenlernen. So ist die Prämisse ähnlich zu Java und XML und die Konklusion einer Regel in Java geschrieben. Jess dagegen hat Ähnlichkeit mit *Prolog* und ist damit für den Laien eher etwas mühsamer zu interpretieren. Der gute Community Support und die Nutzung der Software unter einer Open Source Lizenz, sind zudem positiv zu werten.

In Tabelle 1 wurden die aus der Master-Studienordnung extrahierten Regeln semi-formal dargestellt. Abbildung 10 zeigt nun diese Regeln in die Regelsprache Drools übersetzt. Ein *Doppelpunkt* innerhalb einer Objektdeklaration wird für eine Variablenzuweisung verwendet. Ein Komma steht für ein *logisches Und* (*AND*). Regel *Check_successefully_passed_modules* überprüft mittels eines *Existenzquantors exists* aus der Prädikatenlogik, ob es ein Modul *m* gibt, welches dasselbe Modulkürzel besitzt wie ein bereits gebuchtes Modul, wobei das gebuchte Modul den Buchungsstatus *erfolgreich abgeschlossen* besitzen muss. Ist die Prämisse wahr, kommt die Regel aus Abschnitt 4.3 der Studienordnung zum Einsatz, wonach ein Modul nur einmal angerechnet werden darf. Dazu wird das Attribut *possible* dieser Modulinstanz auf *false* gesetzt. Die Regel *Check_accountability* überprüft, ob ein bereits erfolgreich abgeschlossenes Modul für den Abschluss des Masterstudiengangs angerechnet werden kann. Wenn sein Attribut *possible* bereits auf *false* gesetzt ist, heisst das, dass bei der Regel *Check_successefully_passed_modules* die Prämisse bereits erfüllt gewesen ist. Führt auch diese Regel in die Konklusion über, werden beim *Study*-Object die Kreditpunkte von ursprünglich 120 um die Anzahl der ECTS dieses Moduls reduziert. An dieser Stelle wurden zwei regeln weggelassen. Die nächste abgebildete Regel *Check_precondition* überprüft, ob die für die Anrechnung eines Moduls nötige(n) Voraussetzung(en) erfüllt sind. Sollte das Modul m noch buchbar sein (*possible* == *true*), das Attribut

[16] Software, die eine Zeichenkette durch Angabe eines Trennzeichens in mehrere Teilzeichenketten, so genannte *Tokens* aufteilt.
[17] Another Tool for Language Recognition
[18] Document Object Model: API für den Zugriff auf XML-Dokumente
[19] JSR-94 kompatibler Rule Engine-Standard / Sprache in der die Regeln verfasst werden
[20] Weit verbreitete kommerzielle Rule Engine

preCondition einen Wert besitzen und ein gebuchtes Modul existieren, das erfolgreich abgeschlossen wurde, und dasselbe Modulkürzel besitzen wie m, so sind die Voraussetzungen erfüllt.

```
rule "Check_successefully_passed_modules"
      when
            m : Module(mid : id)
            exists(BookedModule(id == mid,
            bookingStatus == ModuleConst.SUCCESSEFULLY_PASSED))
      then
            m.setPossible(false);
end

rule "Check_accountability"
      when
            s : Study()
            m : Module(possible == false,
            niveau == ModuleConst.NIVEAU_MASTER, accountable == true)
      then
            s.decreaseCreditPoints(m.getEcts());
end

rule "Check_precondition"
      when
            m: Module(possible == true, preCondition != null, preCond : preCondition)
            not exists(BookedModule(id == preCond,
            bookingStatus == ModuleConst.SUCCESSEFULLY_PASSED))
      then
            m.setPossible(false);
end

//rule "Search_prefered_modules"
      //when
            //m : Module(possible == true)
            //eval(isPrefered(m))
      //then
            //m.setPrefered(true);
            //m.setScoringValue(SearchEngine.searchForPreferedModules(m));
//end
```

Abbildung 10: Teilmenge der erstellten Regeln in Drools (Datei: rules.drl)

Mit dem Befehl *eval* wird bei Drools ein boolscher Ausdruck evaluiert. Die letzte unten aufgeführte Regel *Search_prefered_modules* wurde auskommentiert. Anfänglich war die Meinung, dass der statistikbasierte Filter Teil des regelbasierten Filters sein könnte. Dies führt jedoch zu fehlerhaften Statistiken, namentlich dem RSV, der in Abschnitt 3.3.3 eingeführt wird. Die Argumentation für die Auslagerung des statistikbasierten Filters aus dem regelbasierten ist am Ende der Ausführungen zu RSV zu entnehmen.

Die Beschreibung der Funktionsweise des Expertensystems, insbesondere des RETE[21]- Algorithmus, ist nicht Teil dieser Arbeit, da es dazu zahlreiche Literatur

[21] Effizienter Pattern Matching-Algorithmus für Expertsysteme

gibt (beispielsweise [Harmon 1985]). Um die Implementierung der Klasse *RuleEngine* ausreichend zu verstehen, wird empfohlen, das Benutzerhandbuch dieser Software [Proctor et al. 2006] zu studieren.

Die Notwendigkeit eines Expertensystems für den regelbasierten Filter ist nicht per se gegeben. Die hoch optimierten Algorithmen eines heutigen Expertensystems werden dazu verwendet, riesige Datenmengen in kürzester Zeit verarbeiten zu können. Sie sind zur Laufzeit schneller als ein äquivalenter Block von kaskadierten IF-Bedingungen. Diese Unterschiede in der Performance können [Harmon 1985] entnommen werden. Da der Prototyp jedoch nur relativ wenige Moduldaten (beschränkt auf Informatik- und Wirtschaftswissenschaftsmodule) und Regeln verarbeitet, wird ein Unterschied in der Bearbeitungsgeschwindigkeit nicht wahrgenommen. Der Einsatz eines Expertensystems im Empfehlungssystem ist dennoch legitimiert. Zum einen, weil eine operative Lösung das gesamte Angebot an Modulen des Vorlesungsverzeichnisses in kürzester Zeit verarbeiten sollte. Zum anderen, um die Regellogik von der restlichen Businesslogik des Empfehlungssystems standardisiert trennen zu können. Dies hat zum Ziel, die Wartbarkeit einer Software zu erhöhen. Gibt es eine Änderung in der Studienordnung, soll ein NICHT-Informatiker in der Lage sein, diese Änderung vorzunehmen, ohne dass ein Informatiker zur Hilfe beigezogen werden muss.

3.3.3 Statistikbasierter Filter

Der statistikbasierte Filter sucht nach präferierten Modulen. Durchsucht wird je Modul der Inhalt der Attribute *title* und *description* der Klasse *Module* (Abbildung 8). Die Suchbegriffe sind Freitextpräferenzen (Abschnitt 2.3.4) und kategoriale Präferenzen (Abschnitt 2.3.5). Sie werden in Objekten der Klasse *Preference* gespeichert. Für den Prototyp wurde eine Implementierung einer Suchmaschine eingesetzt, die ein auf *Vektoren basiertes Retrievalmodel* verwendet. Apache Lucene (Anhang E, Software 3) erfüllt dieses Kriterium und ist zudem weit verbreitet. Einer der Vorteile dieses Modells ist der für ein Modul berechenbare *Retrieval-Statuswert* (*retrieval status value, RSV*). Dieser ist, einfach gesagt, ein Mass dafür, wie relevant ein Modul in Bezug auf einen Präferenzbegriff (-term) im Gegensatz zu allen anderen Modulen ist. Ein RSV von 0.8 entspricht einer Relevanz von 80%. Der resultierende Wert je Modul und je Präferenzbegriff liegt zwischen 0 und 1 resp. 0% und 100%. Dadurch lässt sich eine Rangliste (*ranking*) der präferierten (relevanten) Module erstellen. Je höher ein RSV desto eher landet ein Modul in einer Modulgruppe resp. in einem Studienplan.

Da der RSV neben dem Titel eines Moduls in den generierten Studienplänen erscheint und der Benutzer oder zumindest eine Fachperson diesen annähernd interpretieren können sollte, wird nachfolgend kurz in die Theorie von Vektormodellen eingeführt und die Berechnung des RSV betrachtet.

In einem einfachen Vektorraummodell bilden sowohl Dokumente als auch Suchanfragen Vektoren. Ein Vektor eines Dokumentes könnte sich aus der Häufigkeit der einzelnen Worte (*Terme*) zusammensetzen. Formal können diese Vektoren wie folgt dargestellt werden:

$$\vec{d}_j = (tf_{1,j}, tf_{2,j}, tf_{3,j}, ..., tf_{k,j})$$

$$\vec{q}_j = (tf_{1,j}, tf_{2,j}, tf_{3,j})$$

Vektor $\vec{d}_j$ besteht aus den Häufigkeiten der Terme des j-ten Dokuments. $\vec{q}_j$ analog dazu ist die j-te Suchanfrage (*query*), bestehend aus einem oder mehreren Suchbegriffen. Je ähnlicher (*degree of similarity/correlation*) sich diese beiden Vektoren sind, desto eher landet ein Dokument in der Ergebnismenge der Suchanfrage.

Eine mögliche Methode, diese Ähnlichkeit zu quantifizieren, ist die Ermittlung des *tf-idf*-Wertes (einer Art von RSV), welcher ein statistischer Wert zur Gewichtung von Dokumenten im *Informaton Retrieval* ist. *tf* steht für *term frequency* (*Termfrequenz*) und ist ein Wert für die relative Häufigkeit eines Terms innerhalb eines Dokuments, also wie häufig ein Term in Bezug zu allen Termen in einem Dokument vorkommt. *idf* steht für *inverse document frequency* (*inverse Dokumenthäufigkeit*) und ist ein Wert für die Menge aller durchsuchten Dokumente im Verhältnis zur Häufigkeit der Dokumente in denen der gesuchte Term vorkommt. Formal wird der tf-idf-Wert wie folgt berechnet:

$$tf = \frac{n_i}{\sum_k n_k} \quad idf = \log \frac{D}{d_i} \quad tfidf = \frac{n_i}{\sum_k n_k} * \log \frac{N}{n_i}$$

n_i = Häufigkeit des Terms i im Dokument mit k Termen
D = Menge aller Dokumente
d_i = Menge aller Dokumente die den Suchterm enthalten.

(vergleiche [Baeza-Yates & Ribeiro-Neto 1999], [Türker 2007], verändert)

Die Termfrequenz besitzt lokalen, die inverse Dokumentfrequenz globalen Charakter. Je höher beide Werte sind, desto höher ist auch ihr Produkt (tf-idf). Wörter die in allen Dokumenten sehr häufig vorkommen, wie z.B. *Artikel* ergeben eine tiefe inverse Dokumentfrequenz und somit einen tieferen Produktwert.

Die eingesetzte Suchmaschine benutzt eine komplexere Variante dieser Formel, die an dieser Stelle nicht beschrieben wird. Sie kann jedoch, inkl. Beschreibung, der API zur Suchmaschine entnommen werden.

Nachdem nun der RSV eingeführt wurde, wird nun auf die im vorherigen Abschnitt getroffene Designentscheidung der Auslagerung des statistikbasierten Filters aus dem regelbasierten eingegangen. Das Problem ist, dass bei einem Expertensystem jeweils nur ein Objekt (auch *Faktum* genannt) nach dem anderen (in diesem Falle ein Modul) durch die Regeln zeitgleich geprüft wird. Dem statistikbasierten Filter würden somit nur ein Modul pro Zeit übergeben werden. Dessen Inhalt würde durchsucht werden und es würde der RSV aufgrund eines Moduls anstatt der Menge D berechnet werden. Wie aber obige Formel und Ausführungen gezeigt haben, ist der Einbezug der Menge D nötig, um ein korrektes Resultat zu erhalten. Aufgrund dieses Sachverhaltes wurde der statistikbasierte Filter ausgelagert und wird nun nach der regelbasierten Filterung aufgerufen.

Zurzeit werden die Präferenzbegriffe ohne Zusatz von *Wildcards* als Suchbegriffe verwendet. Dies führt natürlich dazu, dass wenn ein Wort in einem Modul nicht exakt gleich geschrieben wurde wie der Suchterm, kein Treffer vorliegt. Führt man jedoch z.B. den Wildcard-Operator „*“ am Ende eines Suchterms ein, so wird der RSV verringert. Als Beispiel dafür kann das Wort „Finance“ genommen werden. Schneidet man am Ende dieses Wortes zwei bis drei Buchstaben (Angabe meiner Betreuerin) weg und fügt man den obigen Operator hinzu, dann erhält man „Finan*“ dadurch werden auch Inhalte als präferiert erachtet, die z.B. das Wort „Financial“ enthalten. Das gekürzte Wort kommt nun in einer grösseren Zahl von Dokumenten (Modulen) vor als zuvor, wodurch aufgrund der Formel zur Berechnung des RSV dieser Suchterm an Relevanz verliert. Dies kann zur Folge haben, dass auch unerwünschte Inhalte als präferiert angezeigt werden. Die Benutzung von Wildcards könnte als optional angeboten werden. Vielleicht wäre es für den Studenten auch wünschenswert, eine ausgeklügeltere Ähnlichkeitsprüfung (*similarity check*) zur Verfügung zu haben. Im Falle, wo Modulinhalte auf Englisch geschrieben sind, die Präferenzangaben jedoch auf Deutsch (oder umgekehrt), könnte die Einbindung eines mehrsprachigen *Lexikons* oder *Thesaurus* ebenfalls gute Dienste leisten.

3.3.4 Studienplan-Generator

Der Studienplan-Generator ist das letzte Glied des Empfehlungssystems. Er hat vom regelbasierten Filter Module erhalten, die möglich, teilweise auch präferiert und absteigend nach ihrem RSV sortiert sind. Wie in Abbildung 5 dargestellt, finden in diesem letzten Glied folgende Prozessschritte statt:

- Modulgruppenbildung
- Aufteilung der Module nach Semestern
- Generierung der Studienpläne

Die Modulgruppenbildung dient der Bildung der in Abbildung 3 dargestellten 4 Modulgruppen. Diese werden für unterschiedliche Kombinationen von Modulen gebildet, um später für die Generierung von unterschiedlichen Studienplänen eingesetzt zu werden. Die Bildung solcher Gruppen ist jedoch kein triviales Problem und wird nachfolgend erläutert. Es werden zwei Alternativen beschrieben, wie dieses Problem gelöst werden kann.

Jede Modulgruppe besitzt ein Minimum und ein Maximum an Kreditpunkten, die erreicht werden müssen resp. angerechnet werden können. Zur Bildung der Modulgruppen wird jedoch primär das Minimum als Obergrenze (*threshold*) berücksichtigt. Eine Modulgruppe muss 3 Kriterien genügen:

1. Je Modulgruppe muss die gesetzte Obergrenze erreicht werden oder sie darf nur möglichst gering überschritten werden.
2. In einer Modulgruppe müssen die Anzahl Module des Herbst- und Frühlingssemesters in einem bestimmten Verhältnis stehen (Attribut semesterType).
3. Die Modulgruppen müssen zu einem bestimmten Prozentsatz unterschiedliche Module enthalten.

Kriterium 1 soll mit Abbildung 11 verdeutlicht werden. Darin sind mehrere Module (Modul1-10) und drei Gruppierungsmöglichkeiten, durch Kreise gebildet,

dargestellt. Die Obergrenze der Gruppe ist hier bei 18 ECTS festgelegt. Diese wird jeweils durch genau die Summe der ECTS ihrer Module erreicht.

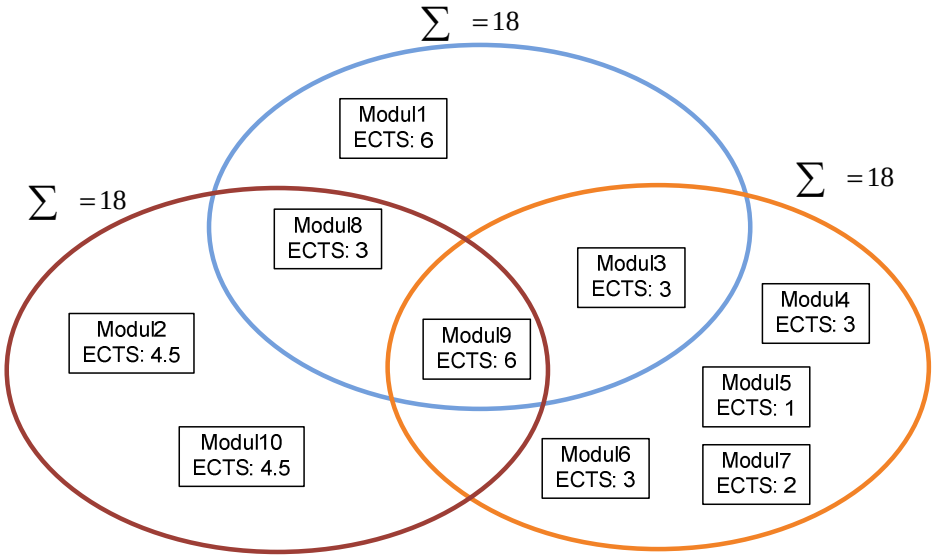

Abbildung 11: Illustration zur Modulgruppenbildung

Werden jedoch zusätzlich *Kriterium 2 und 3* berücksichtigt, ist die Bildung mehrerer solcher Gruppierungsmöglichkeiten oftmals nicht mehr möglich, da zu wenig Daten zur Verfügung stehen, um die Kriterien zu erfüllen. Die Anzahl der Gruppierungsmöglichkeiten hängt von den Kriterien 2 und 3 und von der Anzahl Kreditpunkte je Modul ab. Dadurch ist auch die Menge der Studienpläne begrenzt, denn pro Studienplan wird jeweils eine Modulgruppeninstanz verwendet.

Es wurden 2 Algorithmen (Alternativen) evaluiert, die für obiges Problem eingesetzt werden können:

- Einfacher Kombinationsgenerator (Kombinationen ohne Wiederholung)
- Rucksack-Algorithmus (*knapsack*)

Die erste Alternative berechnet aufgrund der nachfolgenden Formel aus der Kombinatorik (Schulstoff) alle Kombinationen aus n Objekten mit einer Gruppengrösse von k Objekten:

$$\left(\frac{n}{k}\right) = \frac{n!}{k!*(n-k)!}$$

Wird die Datenstruktur mit Modulobjekten dem Kombinationsgenerator übergeben, wird die Anzahl enthaltener Module n ermittelt. Das Resultat ist dann eine Kombination der Grösse k von Indexwerten. Damit kann man auf die Datenstruktur zugreifen und die Module herauslesen.

Dieser Algorithmus ist jedoch relativ komplex sowie ineffizient und skaliert schlecht bei einer grossen Menge von Moduldaten. Ineffizient ist er deshalb, weil zuerst alle möglichen Kombinationen berechnet werden, wobei die meisten gar nicht benötigt werden. Jene Kombinationen, welche alle oben erwähnten

Kriterien für eine Modulgruppe erfüllen, sind darin enthalten und müssen durch einen Filter (*post processing filter*) herausgelesen werden.

Die zweite Alternative ist die Verwendung des Rucksack-Algorithmus (*knapsack*). Dieser kann wie folgt beschrieben werden:

Geg: N Objekte mit Gewicht (*size*) s und Kosten (*profit*) p, und die Kapazität (*capacity*) C des Rucksacks
Ges: Eine Teilmenge T der Objekte, sodass die Summe der Gewichte $\leq$C und die Summe der Kosten maximal ist

Übertragen auf die Bildung der Modulgruppen bedeutet dies: N Module (Objekte) mit einer bestimmten Anzahl Kreditpunkte (Gewichte) und einem möglichst hohen RSV (profit) und Obergrenze (Kapazität) der Modulgruppe (Rucksack).

Die ursprüngliche Implementierung dieses Algorithmus ist in rekursiver Form. Zur Laufzeit ist diese jedoch von der Ordnung O(2^N). Sie gehört daher zur Komplexitätsklasse der *NP-vollständigen* Probleme, welche nichtdeterministisch-polynomial sind. Eine Modulgruppe würde daher nicht in polynomialer Zeit berechnet werden können. Mittels *dynamischer Programmierung*[22] und somit einer Neu-Implementierung dieses Algorithmus, ist das Problem nur noch von der Ordnung O(C*N), wobei C ein konstanter Faktor ist [Cap 1993][Sedgewick 2003]. Sie folgt dem Paradigma *teile und herrsche* (*devide & conquer*), wodurch ein linearer Aufwand resultiert. Aus diesem Grund wird der Rucksack-Algorithmus im Gegensatz zur ersten Alternative präferiert. Er wurde im Prototyp implementiert, jedoch noch mit einer kleinen Anpassung versehen. Die Gewichte (ECTS) sollen kleiner gleich der Kapazität des Rucksacks (Obergrenze) sein. Bei einer Modulgruppe darf aber das Minimum der nötigen Kreditpunkte nicht unterschritten werden. Der Algorithmus wurde deshalb so angepasst, dass er sich in einer kontrollierten Schlaufe befindet. Sollte die Summe der Gewichte kleiner als die initiale Kapazität sein, so wird die Kapazität inkrementiert und der Algorithmus noch einmal ausgeführt, bis eine Kombination gefunden wird, die dieses Kriterium erfüllt, oder ein Schlaufenzähler eine Höchstgrenze (*Abbruchbedingung*) erreicht hat.

Nach der Bildung aller Modulgruppen wird jeweils eine Modulgruppeninstanz genommen, um zusammen mit den Pflichtmodulen (Master-Basismodul, Projektarbeit und Masterarbeit) und dem Wahlpflichtmodul Informatik Seminar die Semester des Studiums möglichst gleichmässig mit ca. 30 ECTS pro Semester (Fall Vollzeitstudium) aufzufüllen. Dazu werden die Module in HS-(Herbstsemester) und FS-(Frühlingssemester) Module aufgeteilt. Danach kommt wieder der Rucksackalgorithmus zum Einsatz. Die Semester bilden die Rucksäcke. In der Implementierung gibt es keine separaten Datenstrukturen für die Semester, sondern es wird jeweils das Attribut *targetSemester* der Klasse Module gesetzt. Die Pflichtmodule werden gesondert behandelt. Sie werden ausgehend von den Belegungsmöglichkeiten der Abbildung 4 in den jeweiligen Semestern platziert. Der Prototyp ist zurzeit so programmiert, dass er diese Module fix für jeden Studienplan gleich platziert:

[22] Teilt, wenn möglich, ein Gesamtproblem in mehrere Teilprobleme auf und löst das Gesamtproblem dadurch viel effizienter.

1. Semester: Master-Basismodul (frei 27 ECTS)
2. Semester: - (frei 30 ECTS)
3. Semester: Projektarbeit (frei 12 ECTS)
4. Semester: Masterarbeit (frei 0 ECTS)

Die frei zu belegenden Kreditpunkte bilden nun die individuellen Rucksackkapazitäten. Wobei diese adaptiv an die verfügbaren HS- und FS-Modulmengen angepasst werden. Dies, weil wenn zuvor eine Modulgruppe gebildet wurde, deren Summe an ECTS grösser als 30 Punkte ist, müssen diese Module nun alle in den jeweiligen Semestern untergebracht werden, da sonst das Minimum einer Modulgruppe (threshold) unterschritten werden könnte. Der Einbezug weiterer Belegungsmöglichkeiten könnte im Prototyp berücksichtig werden. Da es sich jedoch um ein *Empfehlungs*system handelt, könnte man die Belegung weitgehend belassen. Schreibt man beispielsweise die Masterarbeit bevor man sich alles Wissen und Können aus dem Masterstudium angeeignet hat, könnte die Arbeit qualitativ schlechter ausfallen.

Sobald alle Module ihren Semestern zugeordnet worden sind, kann die Datenstruktur mit den Modulobjekten weiterverwendet werden. Der Prototyp gibt Studienpläne auf der Konsole aus. Es wäre jedoch auch möglich diese Daten via Web-Service von einem anderen System abzurufen. Abbildung 12 zeigt einen Ausschnitt der Ausgabe des Studienplan-Generators. *Score* ist ein Synonym für den RSV. Alle Veranstaltungen mit einem Score > 0 werden angezeigt.

[Stundenplan 1 des Masterstudiums in Informatik, Studienrichtung: Master in Software Systems] (Total Punkte: 122)

Präferenzen: ACM: (0): Requirements/Specifications (1): Testing and Debugging (2): Software Architectures (3): COMPUTER-COMMUNICATION NETWORKS (4): MANAGEMENT
OF COMPUTING AND INFORMATION SYSTEMS
FREE: (5): Software Engineering (6): Morphology (7): Programmieren (8): Artificial Intelligence (9): Reengineering

[1.Semester] (Punkte: 31)
- Modul: Software Reengineering (V+Ü) Ects: 4.0 Scores: [(2): 43% (5): 24% (9): 100%]
- Modul: Requirements Engineering II (V+Ü) Ects: 4.0 Scores: [(5): 20%]
- Modul: Nonstandard Datenbanksysteme (V) Ects: 3.0 Scores: [(2): 6% (5): 16%]
- Modul: Morphology and Computation (V) Ects: 3.0 Scores: [(3): 5% (6): 100% (8): 3%]
- Modul: Introduction to Artificial Intelligence (V+Ü) Ects: 6.0 Scores: [(3): 6% (8): 88%]
- Modul: Biologically and Cognitively Motivated Computation (V+Ü) Ects: 3.0 Scores: [(3): 21%]
- Modul: Soziale und Ökonomische Faktoren in der Informatik (V+Ü) Ects: 3.0 Scores: [ ]
- Modul: Praktikum Datenbanksysteme (PR) Ects: 2.0 Scores: [ ]
- Modul: Master-Basismodul in Softwaresysteme, Prof. Burkhard Stiller Ects: 3.0 Scores: [ ]

[2.Semester] (Punkte: 31)
- Modul: Seminar: Dynamic & Distributed Information Systems SS07 (BSc/MSc Studiengang) Ects: 3.0 Scores: [ ]
- Modul: Advanced Software Engineering (V+Ü) Ects: 6.0 Scores: [(2): 41% (5): 86%]
- Modul: Component-Based Software-Engineering (V+Ü) Ects: 3.0 Scores: [(2): 23% (5): 50%]
- Modul: Programmieren mobiler Systeme (V) Ects: 3.0 Scores: [(7): 100%]
- Modul: Practical Artificial Intelligence (V) Ects: 3.0 Scores: [(8): 99%]
- Modul: Problemlösen im Informatikalltag (V) Ects: 3.0 Scores: [ ]
- Modul: Fallstudien zum Informationsmanagement (V+Ü) Ects: 4.5 Scores: [ ]
- Modul: Sicherheit in der Informationstechnik (V) Ects: 3.0 Scores: [ ]
- Modul: Data Warehousing (V) Ects: 3.0 Scores: [ ]

[3.Semester] (Punkte: 27)
- Modul: Objektorientierte Softwareentwicklung (V) Ects: 3.0 Scores: [ ]
- Modul: Multimedia Anwendungen und Praxis (V+Ü) Ects: 3.0 Scores: [ ]
- Modul: Distributed Systems (V+Ü) Ects: 3.0 Scores: [ ]
- Modul: Projektarbeit in Software Systems Ects: 18.0 Scores: [(2): 30% (5): 17%]

[4.Semester] (Punkte: 33)
- Modul: XML und Datenbanken (V) Ects: 3.0 Scores: [ ]
- Modul: Masterarbeit in Softwaresysteme Ects: 30.0 Scores: [ ]

Abbildung 12: Ausgabe des Studienplan-Generators

4 Evaluation

Die Evaluation des Empfehlungssystems erfolgte mehrheitlich auf qualitativer Ebene, wobei folgende Frage im Zentrum stand:

Welchen Mehrwert bietet das automatisierte Verfahren der Studienplan-Generierung im Gegensatz zur herkömmlichen manuellen Erstellung durch einen Studenten?

Um dazu eine mögliche, tendenzielle Antwort zu geben, wurden 3 Studenten angefragt, eine manuelle Studienplanung für ihr Masterstudium in Informatik durchzuführen. Sie mussten dazu folgende Informationen zusammenstellen und einsenden:

- Kopie ihres Leistungsausweises (ohne die Noten!)
- Wahl ihrer Studienrichtung
- Zeiteinteilung (Vollzeitstudium oder Teilzeitstudium)
- Nebenfach (Ja/Nein)
- 5 Freitextpräferenzen für die Modulwahl
- 5 ACM-Kategorien für die Modulwahl
- Studienplan: Alle Semester mit der Modulbelegung in Tabellenform
- Kommentar zur manuellen Studienplangenerierung und der Suche nach Präferenzen
- Zeitaufwand

Die Analyse der erhaltenen Daten wird im Folgenden präsentiert. Die von den Studenten erhaltenen Evaluationsdaten können Anhang C entnommen werden. Die Leistungsausweise der Studenten wurde aus Gründen des Datenschutzes nicht beigefügt. Alle Angaben sind anonymisiert.

In einem ersten Schritt wird das Empfehlungssystem mit den Daten eines Studenten zur Studienrichtung und den Präferenzen zur Modulwahl gefüttert, und es werden die Studienpläne generiert. In einem zweiten Schritt werden der vom Studenten erhaltene und der ähnlichste generierte Studienplan (gleiche Module) miteinander verglichen. Dabei spielt die Einteilung der Module auf die Semester eine untergeordnete Rolle, da die Belegung durch das Empfehlungssystem nur aufgrund der *Voraussetzung* eines Moduls systematisch geordnet ist. Als Vergleichswert wird der Prozentsatz der Anzahl gleicher Module im Verhältnis zur Durchschnittsmenge gewählter Module genommen. Die Durchschnittsmenge berechnet sich aus dem Durchschnitt der Anzahl Module der beiden Studienpläne. Zudem wird geschaut, ob zu den Präferenzangaben überhaupt geeignete Module gefunden werden. Der RSV der Module im generierten Studienplan wird dazu als Mass verwendet. Die Kommentare und der Zeitaufwand werden ebenfalls genauer angeschaut. Zuletzt wird ein Fazit im Hinblick auf die zu Beginn dieses Kapitels formulierte Frage erstellt.

In Tabelle 2 sind die Ergebnisse der Auswertung aufgeführt. Alle Studenten haben als Zeiteinteilung das Vollzeitstudium und kein Nebenfach gewählt. In Anhang C ist je Student der automatisch generierte Studienplan zusammen mit

den Präferenzdaten zu sehen. Zudem wurde in einer abgespeckten Version der manuell erstellte Studienplan hinzugefügt.

Student	Studienrichtung	Zeifaufwand in Std.	% Übereinstimmung	Kommentar(e)
1	Information Systems	2	30	-Passende Freitextpräferenzen zu finden war nicht leicht.
2	Software Systems	2	45	-Fünf Präferenzen je Kategorie sind zu wenig um seine Präferenzen auszudrücken. -Die ACM-Kategorien tönen spannend, jedoch wird vieles davon am IFI nicht angeboten. -Die Zusammenstellung der Informationen fürs Empfehlungssystem dauerte 5 Min., jene für den manuellen Studienplan 2 Stunden. Grund: -VVZ ist unübersichtlich -Studienordnung nicht immer eindeutig -Verbesserungsvorschläge: -Alle Vorlesungen je Fakultät auf einer Seite übersichtlich dargestellt -Verbesserte Übersicht über die Vorbedingungen zu einem Modul.
3	Software Systems	1	44	-Die Kategorisierung von ACM eignet sich nicht zur Beschreibung meiner Wunschinhalte. Die Zusammenstellung der Informationen war mittelschwierig. -Informationen über die Module sind zu stark verteilt: auf der IFI-Homepage, im VVZ und auf der Veranstalterseite. -Zudem fehlen die Punkte teilweise auf der IFI-Homepage. Könnten alle Informationen zentral abgefragt werden, würde dies Arbeit erleichtern.

Tabelle 2: Auswertung der Evaluationsdaten

Student 1 hat sich als einziger für die Studienrichtung *Information Systems* entschieden. Die prozentuale Übereinstimmung zwischen dem manuell erstellten und dem automatisch generierten Studienplan beträgt hier nur gerade mal 30%. Dort, wo es zu Übereinstimmungen gekommen ist (hoher RSV), waren die Präferenzbegriffe präzise genug gewählt worden. Dies spricht für die Funktionstüchtigkeit des Systems und ist in Anhang C ersichtlich. Der Hauptgrund für die Abweichungen zwischen diesen beiden Studienplänen sind zu allgemein formulierte Präferenzbegriffe. Die Freitext-Präferenz „Wirtschaft" beispielsweise kommt natürlich in zahlreichen Modulbeschreibungen vor und hat dadurch einen tiefen RSV zur Folge. Aus dem Kommentar des Studenten ist folglich auch zu entnehmen, dass er Mühe hatte, passende Begriffe zu finden. Ein Grund dafür ist sicherlich das Fehlen einer Kategorisierung der Wirtschaftswissenschaften. ACM deckt lediglich die Computerwissenschaften ab. Ein weiterer Grund könnte eine Orientierungslosigkeit des Studenten sein, sodass er noch gar nicht in der Lage ist, die Fülle an Wissenschaftsbereichen zu kennen, die ihn interessieren könnten. Vielleicht müsste der Student im Vorfeld mit ehemaligen Studierenden sprechen und Berufsfelder in der gewählten

Studienrichtung genauer betrachten, um eine zielorientiertere Studienplanung vornehmen zu können.

Bei *Student 2* ist die Wahl auf die Studienrichtung *Software Systems* gefallen. Die prozentuale Übereinstimmung ist in diesem Falle höher und liegt bei 45%. Im Kommentar erwähnt der Student, dass die geforderte Menge an Präferenzbegriffen nicht ausreichend ist, um seine Wünsche auszudrücken. Die Zahl der Präferenzbegriffe kann natürlich (beliebig) erhöht werden. Die Festlegung einer Mindestanzahl, die höher ist als die von 5 Stück, könnte das Ergebnis weiter verbessern. Der zweite Kommentarpunkt bezieht sich auf die ACM-Kategorien. Diese sind oft sehr allgemein formuliert und somit zu unspezifisch zur Beschreibung der angebotenen Informatikmodule. Viele davon beziehen sich auf Wissenschaftsbereiche, die am IFI gar nicht angeboten werden. Dieser Kritikpunkt stellt dieses Kategoriensystem in Frage. Womöglich wäre es vorteilhafter, wenn die Modulverantwortlichen Suchbegriffe und/oder kurz gehaltene Zusammenfassungen anbieten, die viel präziser den Inhalt von Modulen beschreiben würden. Der dritte Kommentarpunkt betrifft die manuelle Erstellung des Studienplans. Der Student kritisiert den Aufbau und die Dezentralität der Modulinformationen. Dies spricht für den Einsatz eines Empfehlungssystems, welches die nötigen Daten zusammenträgt.

Student 3 hat dieselbe Studienrichtung wie Student 2 gewählt. Der Prozentsatz der Übereinstimmung ist beinahe derselbe wie von Student 2. Speziell bei diesem Studenten ist jedoch die Missachtung einer Regel aus der Studienordnung. Wie aus Anhang C entnommen werden kann, hat er zwei Module ausgewählt, die nicht auf Masterniveau sind („Corporate Finance I“ und „Banking: Entwicklung von Bankprodukten“). Der regelbasierte Filter lässt diese Wahl nicht zu, und es werden stattdessen andere Module gewählt. Dies führt unweigerlich zu einer Verschlechterung der prozentualen Übereinstimmung. Auch er kritisiert, wie bereits Student 2, das ACM-Kategoriensystem und betont die Schwierigkeiten der manuellen Studienplan-Generierung.

Aufgrund dieser Ergebnisse und in Bezug auf die einleitend gestellte Frage, können folgende Schlüsse gezogen werden. Die manuelle Erstellung eines Studienplans ist keine triviale Aufgabe. Zum einen werden die Reglemente nicht von jedem Studenten auf Anhieb gleich verstanden und zum anderen ist das Zusammentragen der nötigen Informationen zu den Modulen mit Aufwand verbunden. Ein gewisses Mass an Orientierungslosigkeit im Hinblick auf die gewählte Studienrichtung kommt auch noch hinzu. Bei der automatisierten Erstellung des Studienplans durch das Empfehlungssystem werden diese Nachteile zum Teil massiv verringert. Die reine Erstellungszeit für einen Studienplan wird um ein Vielfaches gesenkt. Eine Fehlinterpretation der Reglemente durch den Studenten hat keinen Einfluss mehr auf den resultierenden Studienplan. Die Präferenzangaben auf der anderen Seite müssen von ausreichender Quantität und Qualität sein, damit ein akzeptables Ergebnis zu Stande kommt. Jedoch ist zu sagen, dass auch wenn nicht immer eine exakte Übereinstimmung der Module zu Stande kommt, so fällt eine inhaltliche Übereinstimmung auf Basis von ähnlichen Modulen womöglich gar nicht schlecht aus. Dies wurde jedoch noch nicht genau untersucht. Die Ersetzung der ACM-Kategorien zugunsten von präziseren, dem Angebot besser

angepassten Auswahldaten sollte für eine operative Lösung ins Auge gefasst werden.

Es steht jedoch im Raum, ob es einem Studenten genügt, seine Präferenzen zu einer Studienrichtung mit Begriffen auszudrücken resp. davon, im Falle von Kategorien, inspiriert wird. Nur schon die vollständigen Beschreibungen zu den Modulen können mehr inspirieren als ein Begriff oder der Titel der Veranstaltung. Vielleicht ist ein Student auch nicht nur an „inhaltlicher Optimierung“ des Masterstudiums interessiert. Als Kriterien könnte die Punkteoptimierung (Module absolvieren, die möglichst viel ECTS-Punkte bringen) oder die Möglichkeit, Zeitrestriktionen anzugeben (Anpassung der Veranstaltungen an einen Stundenplan), ins Auge gefasst werden. Ein Export der Daten in z.B. Excel zur weiteren Bearbeitung/Ausbau des Studienplans wäre ein weiteres nützliches Feature.

5 Beschränkungen

Dadurch dass das Empfehlungssystem, sollte es operativ eingesetzt werden, stark abhängig von seinem Systemkontext ist, wurden einige Beschränkungen festgestellt, die im Folgenden erläutert werden. Veränderungen im Systemkontext können die Funktionstüchtigkeit des Empfehlungssystems zum Teil massiv beeinträchtigen.

Bei der Entwicklung des Prototyps wurden aktuelle Moduldaten aus dem Vorlesungsverzeichnissystem eingesetzt. Grund dafür ist die frühe Erkennung von Konsistenzproblemen zwischen der Implementierung (SOLL) und den von aussen fix gegebenen Daten (IST). Der Einsatz von Dummy-Daten hätte gewisse Probleme nicht zum Vorschein gebracht.

Eine Erkenntnis daraus war, dass das Empfehlungssystem stark von der Quantität und der Qualität der Moduldaten abhängig ist. In quantitativer Hinsicht geht es darum, Kriterium 3 (*Die Modulgruppen müssen zu einem bestimmten Prozentsatz unterschiedliche Module enthalten*) aus Abschnitt 3.3.4 erfüllen zu können. Wobei auch Kriterium 2 (*In einer Modulgruppe müssen die Anzahl Module des Herbst- und Frühlingssemesters in einem bestimmten Verhältnis stehen*) in gewisser Hinsicht Abhängigkeit von der Menge der Moduldaten mit sich bringt. Je besser die zur Verfügung stehenden Moduldaten diese Kriterien erfüllen, desto mehr unterschiedliche Studienpläne können erstellt werden.

In qualitativer Hinsicht wurden folgende Beschränkungen entdeckt:

1. Die Module müssen Kriterium 2 aus Abschnitt 3.3.4 erfüllen können
2. Konflikte mit dem Trennzeichen
3. Konsistenz der Moduldaten: Voraussetzung
4. Verwendung heutiger Daten für zukünftige Semester

Die 1. Beschränkung ist erfüllt, wenn genügend Module in ausgeglichener Menge zur Verfügung stehen, die Kriterium 2 erfüllen.

Die 2. Beschränkung hat damit zu tun, dass bestimmte Mängel bei den Daten festgestellt wurden, welche die maschinelle Verarbeitung erschweren. Der Parser gibt bei der Verwendung der VVZ-Exporte ohne Überarbeitung Fehler aus. Das Problem ist, dass als Trennzeichen der Daten ein Doppelhochkomma eingesetzt wurde. Dieses Zeichen wird aber auch häufig in den Titeln der Module verwendet, um Begriffe hervorzuheben. Dies bedarf im Vorfeld einer manuellen Überarbeitung. Als Lösung zur Automatisierung bietet sich an, das Trennzeichen zu ändern (z.B. auf „;“) oder ein Direktzugriff auf das Vorlesungsverzeichnissystem zuzulassen.

Die 3. Beschränkung bezieht sich auf folgende Probleme:

- Wünschenswert wäre, wenn das Attribut *Voraussetzung* (eine) Referenz(en) in Form eines oder mehrerer Modulkürzel enthalten würde. Dieser Sachverhalt wurde bereits in Abschnitt 2.3.1 genauer erläutert.

Für den Prototypen mussten die erhaltenen Moduldaten dahingehend korrigiert werden, dass die Regel zur Prüfung der Voraussetzung eingesetzt werden konnte.

- Ein weiteres Konsistenzproblem ist, dass in den VVZ-Exporten zum Teil nicht angegeben war, wann ein Modul angeboten wird (HS od. FS).
- Die Modulkürzel der Unterrichtsassistenzen enthalten das Niveaukürzel fürs Bachelorniveau, im Bereich *Bestandteil von* werden sie jedoch auch für Masterstudierende angeboten. Systemmässig ergibt sich dieselbe Problematik wie beim Inhalt des Attributs *Voraussetzung*.
- Im Hinblick auf die Ausweitung des Systems ist anzumerken, dass der Aufbau des Modulkürzels abhängig von der Fakultät ist. So weisen beispielsweise Module aus den Publizistikwissenschaften unterschiedlich aufgebaute Modulkürzel auf.

Die 4. Beschränkung adressiert eine generelle Beschränkung des Empfehlungssystems. Da im optimalen Fall die Moduldaten nur 2 Semester im Voraus bekannt sind (aktueller Kenntnisstand), das Empfehlungssystem jedoch über alle Semester des Masterstudiums die Studienpläne im Voraus erstellt, ist es möglich, dass ein Modul zwischenzeitlich gar nicht mehr angeboten wird. Da das Empfehlungssystem aber nicht verbindliche, sondern empfohlene und möglichst realistische Studienpläne ausgibt, erfüllt es dennoch seinen Zweck.

6 Künftige Arbeiten

Notwendige und wünschenswerte Anforderungen, die in einer operativ einsetzbaren Lösung des Empfehlungssystems enthalten sein müssen/sollen, jedoch nicht Teil dieser Arbeit sind, werden nachfolgend erläutert.

6.1 Zusätzliche Features

Die in dieser Arbeit erwähnten noch nicht implementierten Features sind nachfolgend nochmals aufgelistet:

- Ähnlichkeitsprüfung (Abschnitt 3.3.3)
- Mehrsprachiges/r Lexikon/Thesaurus (Abschnitt 3.3.3)
- Punktemaximierung (Kapitel 4)
- Zeitrestriktionen (Kapitel 4)
- Datenexport (Kapitel 4)

6.2 Einbettung ins Projekt MyMaster@IFI

Das Empfehlungssystem ist ein Bestandteil des Projekts MyMaster@IFI. Es bildet einen Teil der Geschäftslogik. Dieses Projekt sieht vor, eine *Student Self Service*-Anwendung zu schaffen, die über den geplanten Funktionsumfang des vorgestellten Empfehlungssystems hinausgeht. Es ist beispielsweise geplant, dass sich der Student konkrete Wochenstundenpläne erstellen kann. Zudem soll die Datenbank des Zürcher Verkehrsverbunds (ZVV) eingebunden werden, damit ein Student auch Präferenzen bezüglich Ankunfts- und Abfahrtszeiten bekannt geben kann und danach die Stunden- und Studienpläne ausgerichtet werden.

6.3 Zugriffsbestimmungen und Unterstützung

Damit zukünftig die in dieser Arbeit beschriebenen Beschränkungen reduziert werden können, sind Direktzugriffe auf das Vorlesungsverzeichnis- und das Modulbuchungssystem nötig. Direkte Systemzugriffe wurden für diese Arbeit aus Gründen der Informatiksicherheit und Aspekten des Datenschutzes nicht gewährt. Bei einer Umsetzung des Projekts währe eine Abklärung vorteilhaft.

7 Schlussfolgerungen

Die Ausführungen in dieser Arbeit haben gezeigt, dass ein Empfehlungssystem zur Generierung von Studienplänen umsetzbar ist. Aufgrund der Spezifikation und des Entwurfs konnte ein funktionstüchtiger Prototyp mit einer hohen Abdeckung der funktionalen Anforderungen erstellt werden. Aufgrund der Designentscheidungen wurden auch nicht-funktionale Anforderungen wie Performanceverbesserungen erreicht. Dies durch die Wahl eines Expertensystems für den regelbasierten Filter, das einen effizienten Algorithmus zur raschen Filterung der Daten einsetzt, durch die Wahl einer effizienten Suchmaschine für den statistikbasierten Filter und durch einen effizienten Algorithmus (Rucksack-Algorithmus) zur Erstellung der Studienpläne.

Der Mehrwert dieses Systems gegenüber der manuellen Studienplanerstellung ist die konsequente Einhaltung der Reglemente des Masterstudiums in Informatik, der zeitliche Aspekt und die Möglichkeit, Präferenzen zur Modulwahl einzugeben und/oder aus dem ACM-Kategoriensystem auszuwählen. Der Nutzen von letzterem ist jedoch umstritten. Besser wäre wohl, wenn die Modulverantwortlichen die nötigen Informationen (Begriffe, Kurzzusammenfassungen) zu ihren Modulen bereitstellen würden. Zusätzliche Features sollen das Empfehlungssystem verbessern und somit die Bedürfnisse der Studierenden noch besser abdecken. Dazu gehören unter anderem auch die Möglichkeit der Punktemaximierung und die Angabe von Zeitrestriktionen.

Damit dieses System reibungslos funktionieren kann, sind die in Kapitel 5 dargelegten Beschränkungen hinsichtlich der Qualität und Quantität der Eingabedaten zu berücksichtigen. Ansonsten funktioniert das System nach dem Prinzip „Garbage in, garbage out!“. Um die Verarbeitung der Daten aus dem Vorlesungsverzeichnissystem und dem Modulbuchungssystem zu erleichtern, sind Direktzugriffe auf die Systeme wünschenswert.

7.1 Reflexion des Erreichten

Die Erarbeitung dieser Bachelorarbeit war lehrreich und spannend. Lehrreich, weil an zahlreichen Stellen in dieser Arbeit das erlernte Wissen aus dem Studium nochmals vertieft werden musste und praxisnah umgesetzt werden konnte. Spannend, weil Teile des Systems erst am Prototyp erprobt werden mussten, bevor eine Designentscheidung getroffen werden konnte. Ich hoffe, dass ich mit dieser Arbeit die Verantwortlichen des MyMaster@IFI-Projekts inspirieren konnte und dieses Projekt ein voller Erfolg wird.

Literaturverzeichnis

Baeza-Yates, R., Ribeiro-Neto, B., Modern Information Retrieval, Pearson Education Limited, 1999.

Cap, C., Theoretische Grundlagen der Informatik, Springer-Verlag, Wien, 1993.

Harmon, P., Expert systems – artificial intelligence in business, John Willey & Sons, New York, 1985.

Glinz, M., Gall, H., Software Engineering, Vorlesungsunterlagen, Universität Zürich, 2005.

Proctor M. et al, JBoss Rules User Guide, 3.0.6, JBoss Inc., 2006, http://labs.jboss.com/jbossrules/docs/index.html.

Russell, S. Norvig, P., Artificial Intelligence – A Modern Approach, Pearson Education, Second Edition, New Jersey, 2003.

Sedgewick, R., Algorithmen in Java – Teil 1-4, 3. Auflage, Pearson Education, München, 2003.

Türker, C., XML und Datenbanken, Vorlesungsunterlagen, ETH Zürich, 2007.

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

A ACM-Kategorien (reduzierte Version)

C

- C.2 COMPUTER-COMMUNICATION NETWORKS
  - C.2.1 *Network Architecture and Design*
  - C.2.2 *Network Protocols*
  - C.2.3 *Network Operations*
  - C.2.4 *Distributed Systems*
  - C.2.5 *Local and Wide-Area Networks*
  - C.2.6 *Internetworking*
- C.3 SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS

D

- D.1 PROGRAMMING TECHNIQUES
  - D.1.4 *Sequential Programming*
  - D.1.5 *Object-oriented Programming*
- D.2 SOFTWARE ENGINEERING
  - D.2.1 *Requirements/Specifications*
  - D.2.2 *Design Tools and Techniques*
  - D.2.3 *Coding Tools and Techniques*
  - D.2.4 *Software/Program Verification*
  - D.2.5 *Testing and Debugging*
  - D.2.6 *Programming Environments*
  - D.2.8 *Metrics*
  - D.2.9 *Management*
  - D.2.10 *Design*
  - D.2.11 *Software Architectures*
  - D.2.12 *Interoperability*
  - D.2.13 *Reusable Software*
- D.3 PROGRAMMING LANGUAGES
  - D.3.1 *Formal Definitions and Theory*
  - D.3.3 *Language Constructs and Features*
  - D.3.4 *Processors*
- D.4 OPERATING SYSTEMS
  - D.4.1 *Process Management*
  - D.4.2 *Storage Management*
  - D.4.3 *File Systems Management*
  - D.4.4 *Communications Management*
  - D.4.5 *Reliability*
  - D.4.6 *Security and Protection*

E

- E.1 DATA STRUCTURES
- E.2 DATA STORAGE REPRESENTATIONS
- E.3 DATA ENCRYPTION
- E.4 CODING AND INFORMATION THEORY
- E.5 FILES

F

- F.1.3 *Complexity Measures and Classes*
- F.2.1 *Numerical Algorithms and Problems*
- F.4.3 *Formal Languages*

G

- G.1.1 *Interpolation*

G.1.2 *Approximation*
G.1.3 *Numerical Linear Algebra*
G.1.6 *Optimization*
G.2.2 *Graph Theory*
G.3 PROBABILITY AND STATISTICS

H

H.1 MODELS AND PRINCIPLES
H.1.1 *Systems and Information Theory*
H.1.2 *User/Machine Systems*
H.2 DATABASE MANAGEMENT
H.2.1 *Logical Design*
H.2.2 *Physical Design*
H.2.3 *Languages*
H.2.4 *Systems*
H.2.5 *Heterogeneous Databases*
H.2.6 *Database Machines*
H.2.7 *Database Administration*
H.2.8 *Database Applications*
H.3 INFORMATION STORAGE AND RETRIEVAL
H.3.1 *Content Analysis and Indexing*
H.3.2 *Information Storage*
H.3.3 *Information Search and Retrieval*
H.3.4 *Systems and Software*
H.3.5 *Online Information Services*
H.4 INFORMATION SYSTEMS APPLICATIONS
H.4.1 *Office Automation*
H.4.2 *Types of Systems*
H.4.3 *Communications Applications*
H.5 INFORMATION INTERFACES AND PRESENTATION (e.g., HCI)
H.5.1 *Multimedia Information Systems*
H.5.2 *User Interfaces*
H.5.3 *Group and Organization Interfaces*
H.5.4 *Hypertext/Hypermedia*
H.5.5 *Sound and Music Computing*

I I.2 ARTIFICIAL INTELLIGENCE
I.2.4 *Knowledge Representation Formalisms and Methods*
I.2.6 *Learning*
I.2.7 *Natural Language Processing*
I.2.9 *Robotics*
I.2.11 *Distributed Artificial Intelligence*
I.3 COMPUTER GRAPHICS
I.3.5 *Computational Geometry and Object Modeling*
I.3.7 *Three-Dimensional Graphics and Realism*
I.4 IMAGE PROCESSING AND COMPUTER VISION
I.4.2 *Compression (Coding)*

K K.3 COMPUTERS AND EDUCATION
K.3.1 *Computer Uses in Education*
K.4 COMPUTERS AND SOCIETY
K.4.2 *Social Issues*
K.4.3 *Organizational Impacts*
K.6 MANAGEMENT OF COMPUTING AND INFORMATION SYST.
K.6.1 *Project and People Management*
K.6.5 *Security and Protection*
K.7 THE COMPUTING PROFESSION
K.7.4 *Professional Ethics*

B Extrahierte Regeln (Fortsetzung)

Abschnitt*	Regelname	when	then
3.3		*Präferenz = Vollzeitstudent (oder Teilzeitstudium)*	*Anzahl ECTS im Mittel = 30/Sem. (15/Sem.)*
4.3, 5.10.1		Modul = Masterarbeit AND Anzahl Fehlversuche < 2 AND Semester > = 2	Modul buchbar
4.3		Modul AND NOT zulässige Anzahl Fehlversuche je Modul <= Anzahl Fehlversuche beim Modul AND NOT Anzahl Fehlversuche im Master >= Anzahl zulässige Fehlversuche im Master AND Modul im Lehrangebot	Modul buchbar
5.1		*Präferenz Nebenfach (30 ECTS)*	*Minimal zu erwerbende Anzahl ECTS -= ECTS vom Nebenfach*
5.1		*Modul passt zu Informatik-Wahlpflichtmodule AND ((Anzahl_Punkte_Belegt(Informatik-Wahlpflicht-Modul) + ECTS(Modul)) <= Min_Zu_Erwerben(Informatik-Wahlpflicht-Modul)*	*Modul buchbar*
5.1		*Modul passt zu Informatik-Wahlmodule AND ((Anzahl_Punkte_Belegt(Informatik-Wahlmodule) + ECTS(Modul)) <= Min_Zu_Erwerben(Informatik-Wahlmodule)*	*Modul buchbar*
5.9		*Modul passt zu Freie-Wahlmodule AND ((Anzahl_Punkte_Belegt(Freie-Wahlmodule) + ECTS(Modul)) <= Min_Zu_Erwerben(Freie-Wahlmodule)*	*Modul buchbar*
5.3		*Modul = Master-Basismodul AND Semester(Buchungssemester) = 1*	*Modul empfehlen*
5.3		*Modul = Master-Basismodul AND Semester(Buchungssemester) = 2*	*Modul buchen (Pflicht; letzte Chance)*
5.7		*Modul.Modultyp = Seminar AND Semester(Buchungssemester) < 4 AND (Anzahl_Punkte Beleg(Seminare) + ECTS(Modul) > Max_anrechenbar(Seminare))*	*Modul empfehlen*
5.7		*Modul.Modultyp = Seminar AND Semester(Buchungssemester) = 4 AND Anzahl_Punkte_Belegt(Seminare) = Min_Zu_Erwerben(Seminare))*	*Modul buchen (Pflicht; letzte Chance)*
5.5		*Modul = Projektarbeit AND Erfolgreich_Abgeschlossen(Master-Basismodul)*	*Modul buchbar*
5.8		*Modul = Unterrichtsassistenz AND (Anzahl_Punkte_Belegt(Unterichtsassistenzen) + ECTS(Modul) <= Max_Zu_Erwerben(Unterrichtsassistenz))*	*Modul buchbar*
5.9		Modul.Modultyp = Sprachkurs AND (Anzahl_Punkte_Belegt(Sprachkurse) + ECTS(Modul) < = Max_anrechenbar(Sprachkurs))	Modul buchbar
5.9		*Modul.Modultyp = Tutorat (in Regel Check_master_niveau berücksichtigt)*	*Modul nicht anrechenbar*
5.9		*Studiengang = Wirtschaftsinformatik AND Modul.Genre = Wirtschaftswissenschaften AND Anzahl_Punkte_Belegt(VLs in Wirtschaftswissenschaften) + Modul <= Min_Zu_Erwerben(VLs in Wirtschaftswissenschaften) AND Semester(Buchungssemester) <=4*	*Modul buchbar*
5.9		*Studiengang = Wirtschaftsinformatik AND Modul.Genre = Wirtschaftswissenschaften AND Anzahl_Punkte_Belegt(VLs in Wirtschaftswissenschaften) + Modul <= Min_Zu_Erwerben(VLs in Wirtschaftswissenschaften) AND Semester(Buchungssemester) = 4*	*Modul buchen*
5.9, 7.2		Modul.Universtiät <> UZH AND Zulässige Anzahl überschritten (Mind. 48 Punkte an UZH (davon 36 in Informatik))	Modul nicht buchbar

C Arbeitsdaten zur Evaluation

Student 1

[Stundenplan 1 des Masterstudiums in Informatik, Studienrichtung: Master in Information Systmes] (Total Punkte: 122)

Präferenzen:
ACM: (0): COMPUTER-COMMUNICATION NETWORKS (1): PROBABILITY AND STATISTICS (2): Office Automation (3): ADMINISTRATIVE DATA PROCESSING (4): MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS (5):
FREE: Informationsmanagement (6): Datenbanktechnik (7): Wirtschaft (8): Finance (9): Verteilte Systeme

[1.Semester] (Punkte: 32)
- Modul: Informationsmanagement (V+Ü) Ects: 6.0 Scores: [(5): 100%]
- Modul: Peer-to-Peer Systems and Applications (V+Ü) Ects: 6.0 Scores: [(9): 43%]
- Modul: Biologically and Cognitively Motivated Computation (V+Ü) Ects: 3.0 Scores: [(0): 20%]
- Modul: Nonstandard Datenbanksysteme (V) Ects: 3.0 Scores: [(9): 7%]
- Modul: Soziale und Ökonomische Faktoren in der Informatik (V+Ü) Ects: 3.0 Scores: [ ]
- Modul: Software Reengineering (V+Ü) Ects: 4.0 Scores: [ ]
- Modul: Requirements Engineering II (V+Ü) Ects: 4.0 Scores: [ ]
- Modul: Master-Basismodul in Wirtschaftsinformatik, Prof. Abraham Bernstein Ects: 3.0 Scores: [ ]

[2.Semester] (Punkte: 32)
- Modul: Seminar: Dynamic & Distributed Information Systems SS07 (BSc/MSc Studiengang) Ects: 3.0 Scores: [ ]
- Modul: Data Warehousing (V) Ects: 3.0 Scores: [(3): 97% (9): 10%]
- Modul: Ringvorlesung UNI/ETH: Informationsmanagement (V) Ects: 2.0 Scores: [(5): 95%]
- Modul: Fallstudien zum Informationsmanagement (V+Ü) Ects: 4.5 Scores: [(5): 84%]
- Modul: Sicherheit in der Informationstechnik (V) Ects: 3.0 Scores: [(9): 14%]
- Modul: Programmieren mobiler Systeme (V) Ects: 3.0 Scores: [(9): 64%]
- Modul: Data Mining zur Wissensgewinnung aus Datenbanken (V+Ü) Ects: 4.5 Scores: [(3): 16%]
- Modul: Software-Qualität (V+Ü) Ects: 3.0 Scores: [ ]
- Modul: Theory of Banking and Financial Intermediation (V) Ects: 4.5 Scores: [ ]
- Modul: Asset Allocations and Performance Measurement (V) Ects: 3.0 Scores: [ ]

[3.Semester] (Punkte: 22)
- Modul: Strategic Business Development in a Selected IT Market (PR/Block) Ects: 1.0 Scores: [ ]
- Modul: Wirtschaftsordnungen (Blockseminar) Ects: 3.0 Scores: [ ]
- Modul: Projektarbeit in Wirtschaftsinformatik Ects: 18.0 Scores: [ ]

[4.Semester] (Punkte: 36)
- Modul: Steuerlehre und Unternehmensbewertung (S) Ects: 3.0 Scores: [ ]
- Modul: Idea Engineering (S) Ects: 3.0 Scores: [ ]
- Modul: Masterarbeit in Wirtschaftsinformatik Ects: 30.0 Scores: [ ]

Manuell erstellter Studienplan
Human Computer Interaction
Strategic Business Development in a Selected IT Market
Informationsmanagement (V+Ü)
Protokolle für Multimedia-Kommunikation
Nonstandard-Datenbanksysteme
Requirements Engineering I
Distributed Systems
Data Mining zur Wissensgewinnung aus Datenbanken
Fallstudien zum Informationsmanagement
Sprachkurse
Real Options
Insurance Economics
Fundamental Probability and Computer Programming
Markt und Information
Advanced Corporate Finance I
Services- und Operationsmanagement: Fälle und Methoden
Advanced Financial Economics
Führungspsychologie (V)
Seminar Datenbanksysteme

Student 2

[Stundenplan 1 des Masterstudiums in Informatik, Studienrichtung: Master in Software Systems] (Total Punkte: 122)

Präferenzen:
ACM: (0): Software Architectures (1): Interoperability (2): Reusable Software (3): Network Architecture and Design (4): Security and Protection
FREE: (5): Software Engineering (6): Communication Systems (7): Requirement Engineering (8): Database (9): Computer Languages

[1.Semester] (Punkte: 30)
- Modul: Software Reengineering (V+Ü) Ects: 4.0 Scores: [(0): 43% (2): 43% (5): 24%]
- Modul: Requirements Engineering II (V+Ü) Ects: 4.0 Scores: [(5): 20% (7): 35%]
- Modul: Requirements Engineering I (V+Ü) Ects: 2.0 Scores: [(5): 13% (7): 24%]
- Modul: Distributed Systems (V+Ü) Ects: 3.0 Scores: [(6): 66% (9): 6%]
- Modul: Human Computer Interaction (V) Ects: 3.0 Scores: [(0): 5% (2): 5% (3): 8% (5): 12% (6): 4% (7): 5% (9): 64%]
- Modul: Morphology and Computation (V) Ects: 3.0 Scores: [(6): 27%]
- Modul: Informationsmanagement (V+Ü) Ects: 6.0 Scores: [ ]
- Modul: Praktikum Datenbanksysteme (PR) Ects: 2.0 Scores: [ ]
- Modul: Master-Basismodul in Softwaresysteme, Prof. Burkhard Stiller Ects: 3.0 Scores: [ ]

[2.Semester] (Punkte: 32)
- Modul: Seminar: Dynamic & Distributed Information Systems SS07 (BSc/MSc Studiengang) Ects: 3.0 Scores: [(6): 11%]
- Modul: Advanced Software Engineering (V+Ü) Ects: 6.0 Scores: [(0): 42% (2): 42% (3): 6% (5): 86% (7): 34%]
- Modul: Component-Based Software-Engineering (V+Ü) Ects: 3.0 Scores: [(0): 23% (2): 23% (5): 50% (7): 21%]
- Modul: Introduction to Computer Graphics (V) Ects: 3.0 Scores: [(6): 8% (9): 65%]
- Modul: Mobile Communication Syst. (V+Ü) Ects: 4.5 Scores: [(6): 27%]
- Modul: Practical Artificial Intelligence (V) Ects: 3.0 Scores: [(6): 5%]
- Modul: Problemlösen im Informatikalltag (V) Ects: 3.0 Scores: [ ]
- Modul: Fallstudien zum Informationsmanagement (V+Ü) Ects: 4.5 Scores: [ ]
- Modul: Sicherheit in der Informationstechnik (V) Ects: 3.0 Scores: [ ]

[3.Semester] (Punkte: 30)
- Modul: Objektorientierte Softwareentwicklung (V) Ects: 3.0 Scores: [ ]
- Modul: Multimedia Anwendungen und Praxis (V+Ü) Ects: 3.0 Scores: [ ]
- Modul: Protokolle für Multimedia-Kommunikation (V+Ü) Ects: 6.0 Scores: [ ]
- Modul: Projektarbeit in Software Systems Ects: 18.0 Scores: [(0): 31% (2): 31% (5): 17% (6): 18%]

[4.Semester] (Punkte: 30)
- Modul: Masterarbeit in Softwaresysteme Ects: 30.0 Scores: [ ]

Manuell erstellter Studienplan
Protokolle für Multimedia
Mobile Communication Systems
Seminar Softwaresystems
Software Reengineering
Objektorientierte Softwareentwicklung
Requirements Engineering I
Requirements Engineering II
Intelligent Traders Project
Service Oriented Systems Engineering
Data Mining
Distributed Systems

Student 3

[Stundenplan 1 des Masterstudiums in Informatik, Studienrichtung: Master in Software Systems] (Total Punkte: 120)

Präferenzen:
ACM: (0): Distributed Systems (1): Programming Environments (2): SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS (3): Logical Design (4): Design Methodology
FREE: (5): Software Systems (6): Software Development (7): Network (8): Database (9): Programming Languages

[1.Semester] (Punkte: 30)
- Modul: Software Reengineering (V+Ü) Ects: 4.0 Scores: [(5): 33% (6): 36%]
- Modul: Peer-to-Peer Systems and Applications (V+Ü) Ects: 6.0 Scores: [(0): 9% (5): 19%]
- Modul: Strategic Business Development in a Selected IT Market (PR/Block) Ects: 1.0 Scores: [(5): 7% (6): 100%]
- Modul: Distributed Systems (V+Ü) Ects: 3.0 Scores: [(0): 100% (5): 39%]
- Modul: Mobile Systems Lab Course (PR) Ects: 4.0 Scores: [(0): 11% (5): 22%]
- Modul: Human Computer Interaction (V) Ects: 3.0 Scores: [(0): 3% (3): 15% (4): 15% (5): 20% (6): 4%]
- Modul: Soziale und Ökonomische Faktoren in der Informatik (V+Ü) Ects: 3.0 Scores: [ ]
- Modul: Semantic Web Engineering (V+Ü) Ects: 3.0 Scores: [ ]
- Modul: Master-Basismodul in Softwaresysteme, Prof. Burkhard Stiller Ects: 3.0 Scores: [ ]

[2.Semester] (Punkte: 30)
- Modul: Seminar: Dynamic & Distributed Information Systems SS07 Ects: 3.0 Scores: [(0): 44% (5): 16%]
- Modul: Software-Qualität (V+Ü) Ects: 3.0 Scores: [(5): 33% (6): 35%]
- Modul: Advanced Software Engineering (V+Ü) Ects: 6.0 Scores: [(3): 11% (4): 11% (5): 32% (6): 35%]
- Modul: Introduction to Computer Graphics (V) Ects: 3.0 Scores: [(0): 6% (5): 12%]
- Modul: Artificial Life (V+Ü) Ects: 6.0 Scores: [(3): 8% (4): 8%]
- Modul: Problemlösen im Informatikalltag (V) Ects: 3.0 Scores: [ ]
- Modul: Fallstudien zum Informationsmanagement (V+Ü) Ects: 4.5 Scores: [ ]
- Modul: Erfolgreich durch soziale Kompetenz (PR) Ects: 2.0 Scores: [ ]

[3.Semester] (Punkte: 24)
- Modul: Requirements Engineering I (V+Ü) Ects: 2.0 Scores: [ ]
- Modul: Requirements Engineering II (V+Ü) Ects: 4.0 Scores: [ ]
- Modul: Projektarbeit in Software Systems Ects: 18.0 Scores: [(0): 13% (5): 100% (6): 25%]

[4.Semester] (Punkte: 36)
- Modul: Sicherheit in der Informationstechnik (V) Ects: 3.0 Scores: [ ]
- Modul: Data Warehousing (V) Ects: 3.0 Scores: [ ]
- Modul: Masterarbeit in Softwaresysteme Ects: 30.0 Scores: [ ]

Manuell erstellter Studienplan
Requirements Engineering I
Requirements Engineering II
Protokolle für Multimedia-Kommunikation
Nonstandard Database Systems
Human Computer Interaction
Seminar Distributed Systems
Software Qualität
Software Evolution
Component based Software Engineering
Advanced Software Engineering
Service Oriented Systems Engineering
XML und Datenbanken
Software Reengineering
Objektorientierte Software Entwicklung
Semantik Web-Engineering
Führung von IT-Projekten
Corporate Finance I
Banking: Entwicklung von Bankprodukten

D Verzeichnis referenzierter Medien

Medium 1: Dieser Arbeit beigelegte CD-ROM. Sie enthält eine Readme-Datei, eine Installationsanleitung, den Sourcecode des Prototypen, dessen Dokumentation und alle benutzten Frameworks und Libraries.

E Eingesetzte Frameworks und Libraries

Software 1: JDOM, Version 1.0, www.jdom.org

Software 2: JBoss Rules, Version 3.0.6, www.jboss.com/products/rules

Software 3: Apache Lucene, Version 2.2.0, lucene.apache.org