BLK-Verbundprojekt
„Entwicklung eines Leistungspunktesystems
in den Fachbereichen Elektrotechnik und Informatik“
BLK_V2_8/2004 (Dezember 2004)

Konzept und prototypische Realisierung eines flexiblen Prüfungsverwaltungssystems

Barbara Bennemann und Bernd Braßel
Christian-Albrechts-Universität zu Kiel

BLK BUND-LÄNDER-KOMMISSION
FÜR BILDUNGSPLANUNG UND FORSCHUNGSFÖRDERUNG

towards the european higher education area
bologna process

Impressum

Herausgeber:

Prof. Dr. Gerhard Wenke (Projektleiter)
Fachbereich Elektrotechnik und Informatik, Hochschule Bremen
WWW-Adresse des BLK-Verbundprojektes:
http://www.informatik.uni-kiel.de/~blk-lp

Dr. Hans Fleischhack
Department für Informatik, Carl von Ossietzky Universität Oldenburg
WWW-Adresse des Oldenburger Verbundpartners:
http://www.uni-oldenburg.de/blk-lps

AutorInnen:

Dipl.-Inform. Barbara Bennemann
Dipl.-Inform. Bernd Braßel
Institut für Informatik, Christian-Albrechts-Universität zu Kiel

Titelblattgraphik:
Dr. Elke Wilkeit, Department für Informatik, Carl von Ossietzky Universität Oldenburg
Quelle des Kieler Fotoausschnittes aus der Titelblattgraphik: „Foto:KIEL.SAILING CITY-Lübke“

Inhaltsverzeichnis

Zusammenfassung

Im Zuge des Bologna-Prozesses werden an deutschen Hochschulen gestufte und modularisierte Bachelor- und Master-Studiengänge eingeführt. Sie sollen dazu beitragen, das Hochschulwesen in Deutschland zu modernisieren, seine internationale Attraktivität zu steigern sowie die nationale und internationale Mobilität von Studierenden zu fördern. Bei der Genehmigung und Akkreditierung dieser Studiengänge ist grundsätzlich nachzuweisen, dass sie modularisiert und mit einem Leistungspunktesystem versehen sind.

Die Modularisierung von Studiengängen führt zu einem höheren Aufwand in der Erfassung und Verwaltung der Studierenden- und Prüfungsdaten als es bei den herkömmlichen Diplom- und Magister-Studiengängen der Fall ist. Die den modularisierten Studiengängen zu Grunde liegenden Prüfungsordnungen erfordern nicht nur eine Verwaltung sondern auch eine Interpretation der erfassten Studierenden- und Prüfungsdaten entprechend den gegebenen Anforderungen. Die Entwicklung und Umsetzung von EDV-Konzepten zur Bewältigung des erhöhten Aufwandes spielen eine zentrale Rolle bei der Einführung gestufter und modularisierter Studiengänge.

In der vorliegenden Broschüre wird das an der Universität Kiel entwickelte Konzept eines flexiblen Prüfungsverwaltungssystems vorgestellt. Dabei wird auf die Vorzüge der Implementierung mit deklarativen Programmiersprachen eingegangen und es wird die prototypische Realisierung mit Hilfe der Programmiersprache `Curry` vorgestellt. Dieser Prototyp stellt Funktionalität für die Auswertung der Studierenden- und Prüfungsdaten des im Wintersemester 2002/03 eingeführten Bachelor-Studiengangs Informatik zur Verfügung.

Kapitel 1

Prüfungsverwaltungssysteme

Im Zuge des Bologna-Prozesses [3] werden an deutschen Hochschulen gestufte Bachelor- und Master-Studiengänge eingeführt. Die Ziele der Bologna-Erklärung standen Pate für die Zielsetzungen, die Bund und Länder für die Modernisierung des Hochschulwesens in Deutschland und die Stärkung seiner nationalen und internationalen Attraktivität sowie zur Förderung der internationalen Mobilität von Studierenden entwickelt haben [5]. Nach dem Beschluss der Kultusministerkonferenz [4] ist bei der Genehmigung von Bachelor- und Master-Studiengängen grundsätzlich nachzuweisen, dass diese modularisiert und mit einem Leistungspunktesystem versehen sind. Um außerdem die Mobilität von Studierenden zu unterstützen, ist es für die Hochschulen empfehlenswert, die nationale und internationale Austauschbarkeit von Dokumenten über Studienleistungen *auch in elektronischer Form* zu unterstützen.

Wurden bisher Anerkennungsverträge zwischen einzelnen Hochschulen geschlossen, soll eine zumindest europaweit einheitliche Vergabe von Leistungspunkten (Credits) in Zukunft den problemlosen Wechsel der Bildungsstätte gewährleisten. Natürlich erfordert eine derartige Umstellung eine ungleich genauere Dokumentation und Präsentation der verschiedenen Studiengänge. Module legen die einzelnen notwendigen Bausteine fest, die für den erfolgreichen Abschluss eines Studiengangs erforderlich sind. In eigens von den Hochschulen für jeden Studiengang entwickelten Modulbeschreibungen werden wichtige Informationen über die einzelnen Module sowie deren Inhalte und die von den Studierenden zu erreichenden Lernziele festgehalten. Auf diese Art dienen Modulbeschreibungen der Transparenz der Studiengänge. Doch nicht nur eine derartige Dokumentation der Studiengänge muss höheren Anforderungen genügen. Künftig wird es auch notwendig sein, die Leistungen der bzw. des einzelnen Studierenden genau und verlässlich nachzuhalten und jederzeit auf Anfrage zu bescheinigen. Während es bisher mit Vor- und Hauptdiplom bzw. Zwischen- und Magisterprüfung u.ä. bestimmte definierte Zeitpunkte im Verlauf eines Studiengangs gab, zu denen erbrachte Leistungen gesammelt, überprüft und bescheinigt werden konnten, sind die Hochschulen in Zukunft gezwungen, jederzeit rechtskräftige Auskunft über den Stand der Ausbildung geben zu können. Zusätzlich sind solche Auskünfte nicht nur wie bisher für den hochschulinternen Gebrauch gedacht, sondern müssen internationalen Maßstäben gerecht werden.

Die Umstrukturierung auf Bachelor- und Master-Studiengänge beschäftigt zur Zeit jede Hochschule in Deutschland. Vielerorts werden Konzepte erarbeitet, um Modulbeschreibungen der neuen Studiengänge zu definieren, zu erfassen und zu verwalten sowie den erhöhten Aufwand bei der Verarbeitung von Prüfungsdaten zu bewältigen. Die Entwicklung entsprechender elektronischer Software sowie ihre Anbindung an die bestehenden IT-Landschaften der Hochschulen erfordern einen hohen finanziellen Aufwand. Es ist zu befürchten, dass es bei diesen einmaligen Aufwendungen nicht wird bleiben können. Betrachtet man die Softwareunterstützung, mit der bisher studentische Leistungen erfasst werden, so droht jeder Hochschule in Zukunft ein zu-

sätzlicher Aufwand, um den erhöhten Erwartungen gerecht werden zu können. Die benötigten finanziellen Mittel sind im Augenblick nur an den wenigsten Hochschulen vorhanden.

Unser Ziel ist deswegen die Erstellung eines Softwaresystems, das weit über die bisher üblichen Systeme hinaus in der Lage ist, studentische Leistungen aufzunehmen und bei Bedarf zu dokumentieren. Die entscheidende Idee ist dabei, mit Hilfe einer einfachen *Beschreibungssprache* die jeweilige Prüfungsordnung vollständig zu spezifizieren. Solche Spezifikationen sind für den Menschen lesbar, sie können aber insbesondere auch automatisch verarbeitet werden. Während in herkömmlichen Systemen lediglich *statische Daten* wie die Noten der abgelegten Prüfungen gespeichert werden können, ist es in unserem System möglich, auch die *Bedeutung solcher Daten in Bezug auf den gesamten Studiengang* zu erfassen. Gerade solche Informationen erfordern üblicherweise das detailreiche Fachwissen eines Sachbearbeiters im Prüfungsamt. Und dieses Fachwissen ist zudem nicht Studiengang übergreifend, sondern erfordert genaue Sonderkenntnisse für jedes einzelne Fach. Auf der Basis einer *formalen Spezifikation* des jeweiligen Studiengangs können hingegen in unserem System rechtskräftige Auskünfte über den Studienfortschritt *automatisch* erzeugt werden. Die besondere Kenntnis der Prüfungsordnung ist so lediglich bei der Erstellung dieser Spezifikation erforderlich, typischerweise seitens derjenigen, die die Ordnung ausgearbeitet haben. Damit eine Sprache zur Spezifikation von Prüfungsordnungen in der Praxis handhabbar bleibt, ist ein sehr hohes Abstraktionsniveau erforderlich, das besonders in deklarativen Programmiersprachen vorhanden ist (siehe dazu Abschnitt 2.1).

Neben der Spezifikation von Prüfungsordnungen wird das System es auch ermöglichen, Modulbeschreibungen zur inhaltlichen Definition von Studiengängen heranzuziehen. Diese Modulbeschreibungen werden in einem austauschfähigen Format hinterlegt und dienen mehreren Zwecken.

- Sie sollen einer *inhaltlichen* Dokumentation von Studienleistungen dienen. Dies ist besonders wichtig bei der Aushandlung von Anerkennungsvereinbarungen zwischen verschiedenen Hochschulen, wie sie vermutlich auch in Zukunft eine Rolle spielen werden.

- Beim Wechsel eines Studiengangs sind es grundsätzlich inhaltliche Erwägungen, die zur Anerkennung von studentischen Leistungen führen. Die Frage also, ob ein Student in seinem vorangegangenen Studium bereits adäquate Leistungen erbracht hat, um diese in seinem neuen Fach angerechnet zu bekommen, soll auf der Grundlage solcher Beschreibungen einfacher entschieden werden können.

- Das austauschfähige Format dient der Synchronisation unseres Systems mit evtl. vorhandener anderer Verwaltungssoftware. Z.B. könnte ein System für die Verwaltung der einzelnen Lehrveranstaltungen (mit Termin, Raum etc.) dieselben Modulbeschreibungen verwenden wie unser System. Auf diese Weise könnte der erhöhte Aufwand und die größere Fehleranfälligkeit bei der Pflege redundanter Datenbestände vermieden werden.

Die Speicherung der Daten in austauschfähigen Formaten ist ein wesentliches Kernkonzept des flexiblen Prüfungsverwaltungssystems. Während die bisher an Hochschulen eingesetzten Produkte auf eine Geheimhaltung ihrer internen Datenformate bestehen, scheint es uns gerade wegen des erforderlichen internationalen Austausches unumgänglich, die entsprechenden Formate bekannt zu machen. Die Geheimhaltung des Datenformats ist ein Punkt, an denen man erkennen kann, dass die Hersteller der verbreiteten Prüfungsverwaltungssysteme sich der gesteigerten Erwartungen in diesem Bereich noch nicht ausreichend bewusst sind.

1.1 Unterschiede zu herkömmlichen Systemen

Das derzeit am weitesten verbreitete System zur Erfassung und Verwaltung studentischer Leistungen, wird vertrieben durch die Hochschul-Informations-Systeme GmbH, kurz HIS[1]. Die HIS vertreibt ein ganzes Softwarepaket, um die Verwaltung von Hochschulen zu unterstützen. Neben HIS gehören FlexNow![2], das an der Universität Bamberg entwickelt wurde, Campus.Net[3] von der Firma Datenlotsen oder HE&R (Higher Education & Research) von SAP[4] zu Softwaresystemen, die im Studienmanagement eingesetzt werden. Die Softwarepakete von HIS (u.a. HIS-POS und HIS-LFS) bieten dabei den höchsten Funktionsumfang. So unterstützt z.B. neben HIS-POS nur noch FlexNow! eine Umsetzung von Studien- und modularisierten Prüfungsordnungen oder eine Dokumentenerstellung. Bei der Prüfungsverwaltung berücksichtigen HIS-POS, FlexNow! und Campus.Net die zu Grunde liegenden Leistungspunktesysteme und sind z.B. in der Lage, verschiedene Benotungssysteme aufzunehmen. Einen Studiengangwechsel managen dagegen nur HIS-POS und SAP, Informationen über den bisherigen Studienverlauf erteilt zusätzlich nur noch Campus.Net.

Trotz des weitreichenden Umfangs der HIS-Software gibt es doch Anlass zu Verbesserungen. Dies betrifft hauptsächlich die eingeschränkte Flexibilität des Systems. Häufig wünschen sich einzelne Prüfungsämter Änderungen, die sie nicht selbst durchführen können. In einem solchen Fall muss das Prüfungsamt die HIS direkt ansprechen. Aus den verschiedensten Gründen geht die HIS auf solche Wünsche aber nicht ein, wenn sie nicht von *allgemeinem* Interesse sind und in das Softwarepaket für *alle* Hochschulen eingehen können. Es ist wenig Raum für die individuellen Ansprüche einzelner Institutionen.

Während man mit dem Problem der fehlenden Flexibilität bisher noch leben konnte, wird sich die Lage in Zukunft drastisch ändern. Denn wie bereits erläutert wurde, sind mit der Umstellung des Lehrbetriebs auf Bachelor und Master die Anforderungen an die Systeme immens gestiegen. Derzeit ist es noch bei weitem nicht klar, wie eine für alle Hochschulen einheitliche Erfassung studentischer Leistungen aussehen könnte. Es ist deswegen unumgänglich, jeder Institution einen eigenen Spielraum bei der Erfassung einzuräumen. Die Flexibilität des Systems wird künftig eine ungleich größere Rolle spielen. Derzeit gibt es kein System, das diesen Anforderungen gewachsen wäre.

Nach unserer Einschätzung ist die fehlende Flexibilität nicht etwa auf einen unzureichenden Aufwand seitens der Hersteller zurückzuführen, sondern Folge des grundsätzlich verfolgten Ansatzes. Allgemeine logische Regelungen sind in herkömmlichen Programmiersprachen nicht einfach abzubilden. Dies gilt erst Recht für juristische Regelungen, wie sie in Prüfungsordnungen enthalten sind. Hier kann man sich nur darauf verlassen, mit hohem Aufwand einen Vorrat an Regelungen zur Verfügung zu stellen und zu hoffen, dass alle Prüfungsordnungen sich an diesen Vorrat halten werden.[5] Eben aufgrund des hohen Aufwands und der Notwendigkeit mit den Interna des Systems vertraut zu sein, ist es der einzelnen Hochschule kaum möglich, eine abweichende Regelung selbst in das System einzupflegen.

Wir möchten hier hingegen ein Konzept für einen grundsätzlich anderen Ansatz vorstellen. In hohen Programmiersprachen ist es nämlich möglich, logische Regeln sehr viel direkter zu formulieren. Wie in Abschnitt 2 noch ausführlich gezeigt wird, ist es damit auch möglich, die juris-

[1] http://www.his.de

[2] http://flexnow.uni-bamberg.de

[3] http://www.datenlotsen.de/campus/campusnet

[4] http://www.sap.com/germany/industries/highered/index.aspx

[5] Daher sind aus den entsprechenden Kreisen auch schon einmal Aussagen zu hören wie: „Wenn Sie ihre Prüfungsordnung mit diesem System nicht abbilden können, sollten Sie die Ordnung dringend ändern.“ Eine durchaus bedenkliche Entwicklung, wie wir finden.

tischen Regelungen einer Studienordnung in menschenlesbarer Form abzubilden. Durch diesen Ansatz erhoffen wir uns, dem Nutzer eine bisher unerreichte Flexibilität zur Verfügung stellen zu können, da sein Zugang zum System genauso wenig eingeschränkt ist wie der eines Systemprogrammierers. Der Nutzer ist deswegen nicht auf eine handvoll von Regelungen angewiesen, die der Hersteller des Systems ihm vorgibt.

1.2 Definition des Arbeitspakets im BLK-Verbund 2

Mit der Einführung des modularisierten Studiengangs „Bachelor Informatik“ im Wintersemester 2002/03 entstand auch an der Universität Kiel die Notwendigkeit, Informationen über Module elektronisch zu erfassen, zu verwalten sowie zu Informationszwecken im Internet zu präsentieren. Des Weiteren ergab sich für das Prüfungsamt ein deutlich höherer Aufwand, da mit den neuen Studiengängen mehr Studierenden- und Prüfungsdaten erfasst, verwaltet und ausgewertet werden müssen. Dieser Verwaltungsaufwand wird in Zukunft noch steigen, da zum Wintersemester 2005/06 die Einführung eines konsekutiven Masterstudiengangs geplant ist.

Im Verbund 2 des BLK-Verbundprojektes „Einführung von Leistungspunktesystemen in Hochschulen“ (s. dazu auch [1]) wurde von Anfang an die intensive Beschäftigung mit dem Thema als eine Reihe von Arbeitspaketen definiert. Darunter auch die folgenden:

- Entwicklung von Verfahrensstrategien zur Kooperation mit der Verwaltung
  - Entwicklung eines Leistungspunktesystems
  - Entwicklung von Standards zur Akkumulation und zum Transfer von Prüfungsleistungen
  - Standardisierung von Prüfungsverfahren
- Entwicklung und Implementierung des Konzeptes für eine Moduldatenbank zur Erfassung und Verwaltung von Modulbeschreibungen
- Entwicklung und Umsetzung eines Konzeptes für den Austausch von Modulbeschreibungen zwischen Hochschulen
- Entwicklung eines Konzeptes für ein flexibles Prüfungsverwaltungssystem sowie seine prototypische Realisierung

Für die Entwicklung und Realisierung des Konzeptes eines flexiblen Prüfungsverwaltungssystem sollten folgende Punkte bearbeitet werden:

- Eine Anforderungsanalyse durchführen, inwieweit eine Unterstützung in der Verwaltung der Studierenden- und Prüfungsdaten auf der Institutsebene benötigt wird.
- Anbindung an die im Prüfungsamt verwendeten Verwaltungssysteme, um doppelte Datenerfassung und -verwaltung sowie ein Doppelangebot an Diensten zu vermeiden.
- Erschaffung einer lokalen Infrastruktur mit einem Angebot an benötigten Diensten (z.B. Erstellung von Dokumenten).
- Die Datenauswertung soll effizient erfolgen und benutzerfreundlich präsentiert werden (z.B. über Internet-Schnittstellen).

Es war also das Ziel des Teilprojektes, eine Strategie zur Kooperation mit der Verwaltung und ein Konzept zur Unterstützung der Verwaltung zu entwickeln und umzusetzen. Das in dieser Publikation beschriebene flexible Prüfungsverwaltungssystem ist ein Werkzeug, das eine Auswertung von Studierenden- und Prüfungsdaten in Bezug auf eine spezifizierte Prüfungsordnung vornimmt und als eine Ergänzung der schon vorhandenen Softwarelösungen eingesetzt werden kann.

Das an der Universität Kiel entwickelte Konzept wurde prototypisch realisiert und die Softwarelösung stellt die definierten Dienste zur Verfügung, die für die modularisierten Bachelor-, Master- und Diplom-Studiengänge im Fach Informatik eingesetzt werden können.

Kapitel 2

Das Konzept des flexiblen Prüfungsverwaltungssystems

Im vorangegangenen Kapitel wurde dargestellt, dass die bisher verwendeten Prüfungsverwaltungssysteme nicht die erforderliche Flexibilität bieten. Insbesondere die Datenauswertung erfolgt in der Regel nur in Anlehnung an eine festgelegte Prüfungsordnung. In diesem Abschnitt wird eine kurze Übersicht über die verwendete Art von Programmiersprachen gegeben und danach das Konzept des flexiblen Prüfungsverwaltungssystems näher vorgestellt.

2.1 Deklarative Programmiersprachen

Mit Hilfe *deklarativer Programmiersprachen* ist es möglich, Prüfungsordnungen mehr oder weniger direkt in eine Form zu überführen, die vom System verarbeitet werden kann. Damit kann jede Hochschule eventuelle Änderungen an ihrer Prüfungsordnung oder der zugehörigen Modulbeschreibung selbst einpflegen. Deklarative Programmiersprachen bieten dem Anwender ein besonders hohes Abstraktionsniveau, das zu kompakten Programmen führt, die in wesentlich kürzerer Zeit entwickelt werden können. Der kompakte Code ist außerdem sehr viel lesbarer als der herkömmlicher Sprachen, kann deswegen viel einfacher gepflegt und wiederverwendet werden. Die einfache Semantik deklarativer Sprachen erlaubt eine Vielzahl automatischer Programmanalysen, was zu einer ungleich höheren Sicherheit der erstellten Programme führt. Die meisten Fehler, die sich in herkömmlich erstellten Programmen finden, können mit Hilfe deklarativer Sprachen bereits zur Zeit der Übersetzung automatisch erkannt und dann vom Programmierer behoben werden. Die Benutzung deklarativer Sprachen führt somit zu Software, die mehr Sicherheit und Verlässlichkeit bietet.

Im Wesentlichen sind es zwei Gründe, die der allgemeinen Durchsetzung deklarativer Sprachen im Bereich der Softwareentwicklung entgegenstehen: Zum einen verlangt das hohe Abstraktionsniveau einen erhöhten Bildungsstand vom Programmierer. Erfahrungsgemäß ist insbesondere der Umstieg von herkömmlichen (z.B. imperativen) Sprachen mit einigen Problemen verbunden. Der zweite Grund ist in der vergleichsweise geringen Ausführungseffizienz zu suchen. Die Ausführung deklarativ formulierter Programme erfordert häufig größere Ressourcen, also mehr Rechenzeit und Speicher.

Wir betrachten die beiden genannten Problembereiche als unkritisch, wenn es um den Erfolg des hier dargestellten Projekts geht. Der erwähnte erhöhte Bildungsstand ist nur dort erforderlich, wo eine komplexe Anwendung im Ganzen zu programmieren ist. Die Ergänzung einzelner Regelungen für eine Prüfungsordnung ist dahingegen naturgemäß sehr viel einfacher. Der erhöhte Ressourcenbedarf deklarativer Programme fällt bei der Leistung heutiger Computer kaum noch

ins Gewicht, jedenfalls nicht bei der in diesem Projekt angestrebten Applikationsgröße. Dass die schnellere Programmentwicklung heutzutage der Ausführungseffizienz vorgezogen wird, hat in jüngster Zeit eindrucksvoll der weltweit verbreitete Einsatz der objektorientierten Programmiersprache Java bewiesen, mit deren Ausführungsgeschwindigkeit sich deklarative Sprachen durchaus messen können.

2.2 Flexible Spezifikation von Prüfungsordnungen

Für diesen Abschnitt wird der folgende (fiktive) Student als Beispiel dienen:

```
Name: Julius Musterstudent
Geboren am: 12.2.1980
Immatrikuliert am: 01.10.2001
Matrikelnummer: 123456
...
```

Um derartige Daten in einem System verwalten zu können, ist eine Definition einer entsprechenden *Datenstruktur* anzugeben. In deklarativen Programmiersprachen kann das wie folgt aussehen:

```
data StrukturStudent = Student {vorname        :: String,
                                nachname       :: String,
                                geboren        :: Datum,
                                immatrikuliert :: Datum,
                                matrikelnr     :: String,
                                ...}
```

Dabei steht „`String`“ für eine Zeichenkette. Ist nun `s` unser Beispielstudent, so würde nun `nachname s` gleichbedeutend sein mit der Zeichenkette `"Musterstudent"`.

Als nächstes wollen wir einen einfachen Test formulieren, ob ein gegebener Student einen gegebenen Namen hat oder nicht. Dazu definieren wir eine *Funktion* `mitNamen` mit den zwei *Argumenten* `name` und `student` wie folgt:

```
mitNamen name student = (nachname student)==name
```

Dabei ist `"=="` zu lesen als „stimmt überein mit“ während das einzelne `"="` eher ein „ist definiert durch“ bedeutet. Wenden wir diese Funktion auf unseren Beispielstudenten `s` an, so ist `(mitNamen "Schmitz" s)` gleich `False`, also falsch, während `(mitNamen "Musterstudent" s)` gleich `True`, also wahr ist.

Interessanter wird es nun, wenn man nicht einzelne Studenten, sondern *Listen* von Studenten betrachtet. Man kann nämlich die bereits definierte Funktion `mitNamen` unverändert verwenden, um aus einer solchen Liste all diejenigen mit einem bestimmten Namen zu selektieren. In einfacher Weise kann dazu die vordefinierte Funktion `filter` verwendet werden. Der einfache Aufruf

```
filter (mitNamen "Musterstudent") liste
```

liefert bereits das Gewünschte. Der entscheidende Vorteil ist die Erhaltung der Lesbarkeit. Man sieht dem Ausdruck nicht an, dass hier fortgeschrittene Konzepte am Werk sind.[1]

[1] Die Funktion `filter` ist eine *Funktion höherer Ordnung*, weil sie ihrerseits eine Funktion, in diesem Fall `(mitNamen "Musterstudent")`, als Argument nimmt. `(mitNamen "Musterstudent")` ist eine *partielle Applikation*: Eigentlich besitzt `mitNamen` wie oben definiert zwei Argumente, wird aber hier zunächst mit nur einem Argument aufgerufen.

Ein weiteres Mittel, um einfach und übersichtlich Regeln einer Prüfungsordnung wiedergeben zu können, sind die so genannten *Aufzählungstypen*. So gibt es im Modulsystem für den Abschluss „Bachelor im Fach Informatik“ an der Universität Kiel die Modultypen Grundmodul, Aufbaumodul, Wahlpflichtmodul im Fach Informatik, Wahlpflichtmodul im Anwendungsfach und das Projektmodul, in dessen Rahmen die Abschlussarbeit erstellt wird. Zudem kann jeder Student freiwillig beliebig viele weitere Module belegen, deren Noten auf Wunsch unter bestimmten Voraussetzungen auf dem Zeugnis erscheinen (auch wenn sie keinen Einfluss auf die Abschlussnote haben). Entsprechend kann man deklarativ spezifizieren:

```
data Modultyp = GrundModul
              | AufbauModul
              | WahlpflichtInformatik
              | WahlpflichtAnwendung
              | ProjektModul
              | BonusModul
```

Diese Definition kann man lesen: Ein `Modultyp` ist entweder ein `GrundModul` oder ein `AufbauModul` oder ein ...[2]

Mit dieser kleinen Einführung ist es nun möglich, einige Auszüge aus der Studienordnung für den Bachelor Informatik an der Christian-Albrechts-Universität zu Kiel zu spezifizieren.

> Paragraph 5
> (3) Die Leistungspunkte müssen in folgenden Modulen erworben werden:
>
> - acht Grundmodule gemäß Anlage mit insgesamt 55 Leistungspunkten
> - neun Aufbaumodule gemäß Anlage mit insgesamt 63 Leistungspunkten
> - drei Wahlpflichtmodule aus dem Bereich Informatik gemäß Anlage mit insgesamt 21 Leistungspunkten sowie
> - weitere Wahlpflichtmodule aus einem Anwendungsgebiet oder aus einem interdisziplinären Schwerpunkt mit insgesamt 17 Leistungspunkten
> - Ein Projektvorbereitungsmodul und ein Projektmodul mit insgesamt 24 Leistungspunkten.

Mit Hilfe des oben definierten Aufzählungstyps für die verschiedenen Arten von Modulen können die in Paragraph 5 enthaltenen Angaben zum Umfang an Leistungspunkten als einfache und übersichtliche Fallunterscheidung geschrieben werden (analog auch die Anzahl der jeweilig zu belegenden Module):

```
umfangECTS GrundModul            = 55
umfangECTS AufbauModul           = 63
umfangECTS WahlpflichtInformatik = 21
umfangECTS WahlpflichtAnwendung  = 17
umfangECTS ProjektModul          = 24
umfangECTS BonusModul            = 0
```

Der letzte Eintrag vervollständigt die Information, die in der Ordnung nur implizit enthalten ist: Es gibt natürlich keinen Mindestumfang für Bonusmodule.

Ein abschließendes Beispiel soll verdeutlichen, dass sich auch vergleichsweise komplexe Sachverhalte lesbar abbilden lassen. Der folgende Paragraph entstammt ebenfalls der oben genannten Prüfungsordnung.

[2] Das große M in „`GrundModul`“ ist lediglich eine Konvention zur Lesbarkeit, und hätte genauso gut auch anders geschrieben werden können.

Paragraph 6
Zulassung
(2) Die Zulassung zum Projektmodul setzt zusätzlich voraus:

1. die Vorlage des Studienbuches,
2. den Nachweis über die bestandene Prüfung aller Pflichtmodule; die Prüfung eines Aufbaumoduls kann nachgeholt werden,
3. den Nachweis über die bestandene Prüfung von Wahlpflichtmodulen aus einem Anwendungsgebiet oder aus einem interdisziplinären Schwerpunkt im Umfang von mindestens zehn Leistungspunkten.

Der erste Punkt, das Vorliegen des Studienbuches, ist wohl nicht automatisch zu überprüfen – allenfalls sollte der Sachbearbeiter einen entsprechenden Hinweis auf dem Bildschirm erhalten. Die beiden anderen Punkte können hingegen in einer Funktion `istZumProjektZugelassen` spezifiziert werden, die für einen gegebenen Studenten überprüft, ob er das Projektmodul in Angriff nehmen kann oder nicht. Dabei sind Deklarationen der Form „`let erg = f x y in`“ zu lesen als: „Im folgenden sei `erg` das Ergebnis des Aufrufs `f x y`.“

```
istZumProjektZugelassen student
  = let (belegt,fehlt) =
          abdeckung (pruefungen student) studiengang
        belegtWP =
          filter (istModultyp WahlpflichtAnwendung) belegt
    in
    istLeer (filter (istModultyp GrundModul) fehlt)
    ‘und‘ laenge (filter (istModultyp AufbauModul) fehlt) <= 1
    ‘und‘ summeECTS belegtWP >= 10
```

Die Funktion `abdeckung` berechnet zu gegebener Prüfungsliste und Studiengang zwei Listen: Die von dem Studenten bereits erfolgreich belegten Module (`belegt`) und die noch fehlenden (`fehlt`). Wie oben bereits ausgeführt, filtert die Funktion `filter` alle Elemente einer Liste heraus, die ein gegebenes Kriterium erfüllen. `filter (istModultyp WahlpflichtAnwendung) belegt` ist also die Liste derjenigen belegten Module, die vom Typ „`WahlpflichtAnwendung`“ sind. Damit lässt sich die weitere Spezifikation erschließen: Der Student ist zum Projekt zugelassen, wenn kein Grundmodul fehlt (d.h. die Liste aller fehlenden Grundmodule ist leer) und höchstens ein Aufbaumodul fehlt und die Summe der ECTS-Punkte der bereits belegten Wahlpflichtmodule im Anwendungsfach mindestens zehn beträgt.

Wir haben in dieser Art die gesamte Ordnung für den Bachelor Informatik an der Christian-Albrechts-Universität zu Kiel spezifiziert. Die Spezifikationen erlauben insbesondere die Einfügung längerer Kommentare, so dass es auf einfache Weise möglich ist, zu jeder Spezifikation den entsprechenden Originaltext der Prüfungsordnung hinzuzufügen.

2.3 Flexible Wahl der Prüfungsordnung

Im vorangegangenen Abschnitt wurde gezeigt, dass unser Konzept vorsieht, so weit wie möglich Prüfungsordnungen in einer formalen Spezifikation wiederzugeben. Dies ist bereits ein wesentlicher Teil der Flexibilität des Ansatzes, denn im Gegensatz zu herkömmlichen Systemen ist man so nicht gezwungen, sich auf eine Menge vorgegebener Funktionen zu beschränken, sondern hat eine *universelle Programmiersprache* zur Verfügung.

Ein zweiter wesentlicher Aspekt des vorliegenden Konzepts ist die Flexibilisierung in noch eine andere Richtung: In unserem System ist es möglich, die Daten eines Studenten bezüglich *verschiedener* Prüfungsordnungen auszuwerten. Was nämlich normalerweise in einem Prüfungsverwaltungssystem nachgehalten wird, sind ausschließlich *Ist*-Daten, also z.B. neben Angaben zur Person, die faktisch bereits abgelegten Prüfungen. Was aber darüber hinaus insbesondere für die/den Studierenden selbst, aber auch für Studienberater und Statistiken, besonders interessant ist, sind die *Soll*-Daten. Also Angaben, die Fragen beantworten wie

- Wieviele und welche Prüfungen sind noch abzulegen?
- Welche Endnote wird erreicht, wenn diese Prüfung nicht wiederholt wird, und welche kann bestenfalls noch erreicht werden?
- Reichen die erbrachten Leistungen für den Beginn der Abschlussarbeit?

Diese und ähnliche Fragen beziehen sich nicht nur auf das, was der Student bereits faktisch erreicht hat, sondern auch auf die Zielvorgaben der Prüfungsordnung. Sie sind nur dann zu beantworten, wenn zusätzlich zu den Ist-Daten auch die Regelungen der Prüfungsordnung im System erfasst sind. Es ist grundsätzlich möglich, solche *Soll*-Daten auf dieselbe Weise abzuspeichern, wie die Ist-Daten. Jedoch gehen unter Umständen Abhängigkeiten verloren und führen zu redundanten Daten mit der Gefahr von Inkonsistenzen.

Spätestens, wenn verschiedene Prüfungsordnungen in Bezug gesetzt werden müssen, versagt eine gleichzeitige bzw. zusätzliche Speicherung von Soll-Daten jedoch. Denn Fragen wie die folgenden sind gerade in Zeiten des Umbruchs unvermeidbar:

- Wie stark sinken die Anforderungen, wenn das zweite Hauptfach ab jetzt nur noch Nebenfach wäre?
- Die Prüfungsordnung bietet in meiner Situation zwei Alternativen. Ist eine von beiden aufgrund meiner erbrachten Leistungen günstiger?
- Welche Konsequenzen hätte ein Wechsel auf die neue Prüfungsordnung?

Solche Fragen lassen sich von einem Prüfungsverwaltungssystem nur dann beantworten, wenn Ist-Daten (die faktisch abgelegten Prüfungen) und Soll-Daten (die Regelungen der Prüfungsordnung) *konzeptuell getrennt sind*. Abbildung 2.1 soll diese Konzeption verdeutlichen.

In der Mitte von Abbildung 2.1 ist das flexibles Prüfungsverwaltungssystem zu sehen. Wesentlich ist die Trennung der Eingabedaten: Auf der einen Seite die faktisch bereits abgelegten Prüfungen, symbolisiert durch die „Datenbank“, auf der anderen Seite die Spezifikation der Prüfungsordnung (wie im vorangegangenen Abschnitt beschrieben). Weil diese beiden Komponenten nun grundsätzlich getrennt sind, ist es möglich, die Prüfungsordnung auszutauschen und dabei die Daten der abgelegten Prüfungen beizubehalten. So wird es möglich, Fragen zu beantworten, die den Wechsel der Prüfungsordnung betreffen.

2.4 Flexible Zusammenarbeit mit anderen Systemen

Das von uns vorgestellte Konzept soll noch ein weiteres Manko beheben, das mit herkömmlichen Systemen leider häufig verbunden ist: Die fehlenden Möglichkeiten zur Kooperation mit anderen Systemen. Grundlegend für eine solche Kooperation ist die Möglichkeit zum Datenaustausch in einem offenen Format. Das von uns vorgestellte flexible Prüfungsverwaltungssystem

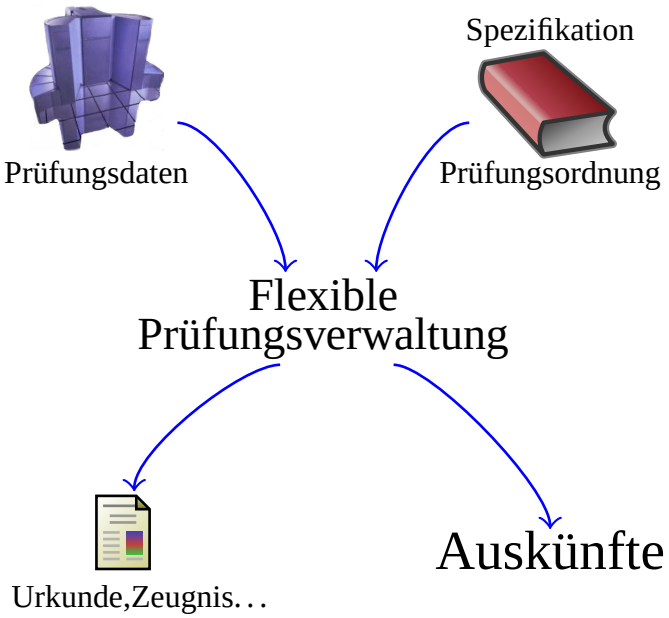

Abbildung 2.1: Konzeptuelle Trennung von Ist- und Soll-Daten

bietet deswegen die Möglichkeit, Daten im XML-Format zu lesen und zu exportieren.[3] Auf diese Weise kann unser System sowohl allein zur Verwaltung von Prüfungsleistungen eingesetzt werden als auch lediglich zusätzliche Funktionalität zu einem bereits vorhandenen System bieten, vorausgesetzt, das vorhandene System unterstützt einen entsprechenden Datenaustausch.

Auf diese Weise wird es möglich, Angaben zu Studierenden aus einer anderen Datenbank, z.B. aus einem Studierendensekretariat, zu erhalten, die Daten für konkrete Veranstaltungen aus einem System zur Verwaltung von Lehrveranstaltungen zu entnehmen und zusätzliche Funktionalitäten zu einem bereits existierenden Prüfungsverwaltungssystem hinzuzufügen. Diese Form der Kooperation unterschiedlicher Verwaltungssysteme erspart die Haltung redundanter Datenbestände und verhindert die ansonsten zwangsläufig auftretenden Inkonsistenzen.

Zusammenfassend: Das Konzept für das flexible Prüfungsverwaltungssystem zeichnet sich durch folgende Eigenschaften aus:

Universelle Beschreibungssprache Eine universelle Spezifikationssprache ermöglicht die mächtige und dennoch lesbare Abbildung von Prüfungsordnungen. Dies erlaubt die Realisierung von Funktionalitäten genau so wie jede einzelne Hochschule sie individuell benötigt und schränkt nicht die Freiheit bei der Erstellung von Prüfungsordnungen ein.

Austausch von Prüfungsordnungen Durch die konzeptionelle Trennung von Ist- und Soll--Daten kann die jeweilige Prüfungsordnung problemlos gewechselt werden. Darüberhinaus wird es möglich, Vergleiche mit denselben Prüfungsdaten aber in Bezug auf verschiedene Prüfungsordnungen vorzunehmen.

Zusammenarbeit mit anderen Softwaresystemen Es ist möglich, Daten mittels offender XML-Formate zwischen schon bestehenden Software-Systemen auszutauschen. Dadurch kann eine problemlose Anbindung an vorhandene Verwaltungssysteme erfolgen.

[3] Einzelheiten dazu in Abschnitt 3.3.

Kapitel 3

Prototypische Realisierung des flexiblen Prüfungsverwaltungssystems

Wir haben eine prototypische Realisierung des im vorangegangenen Kapitels vorgestellten Konzepts erstellt. Der Protyp umfasst:

- Eine Sammlung von vordefinierten Funktionen und Datenstrukturen für die Spezifikation von Prüfungsordnungen.
- Die vollständige Spezifikation von zwei Versionen der Prüfungsordnung des Bachelor-Studienganges Informatik an der Christian-Albrechts-Universität zu Kiel.
- Die automatische Erstellung verschiedener Dokumente wie Urkunde, Zeugnis, Bescheinigung erbrachter Leistungen (Transcript of Records).
- Dienste zur automatischen Abwicklung von Standardanfragen.
- Funktionalität für den Export und Import von Daten im offenen XML-Format.
- Eine Anbindung über dieses Format an mehrere bereits an der Christian-Albrechts-Universität zu Kiel existierende Systeme für die Verwaltung von Studierenden-, Prüfungs- und Moduldaten.
- Eine Internetschnittstelle, um die Erstellung von Dokumenten anzufordern und Anfragen an das System zu stellen.

Die einzelnen Komponenten des Prototyps sollen jeweils kurz in einem eigenen Abschnitt vorgestellt werden.

3.1 Vordefinierte Datenstrukturen und Funktionen

Es gibt eine Reihe spezieller Anforderungen, wenn es um die Spezifikation von Prüfungsordnungen geht. Mit unserem Prototyp existieren Lösungen für viele der grundsätzlich anfallenden Probleme. Im Sinne des vorangegangenen Kapitels ist anzumerken, dass das System aber gerade nicht auf diese vorgefertigten Lösungen eingeschränkt ist. Die Möglichkeit der Spezifikation beliebiger Alternativen macht vielmehr einen wesentlichen Teil der Flexibilität aus.

Im Prototypen enthalten sind Datenstrukturen für:

Grundlegende Typen wie Datum, Noten, ECTS-Punkte. Noten erlauben sowohl die Angabe von Zahlenwerten mit einer Nachkommastelle als auch die einfache Variante Bestanden/-nicht Bestanden. ECTS-Punkte können ebenfalls eine Nachkommastelle aufnehmen.

Studiengänge beinhalten im wesentlichen eine Liste von Modulen. Module sind durch einen Code, Titel und Leistungspunkte definiert.

Modulprüfungen Zu einem gegebenen Modul können mehrere Prüfungen abgelegt werden. Prüfungen sind an einen bestimmten Einzeltermin oder an einen Zeitraum gebunden (z.B. Abschlussarbeiten, Praktika). Zudem sind ihnen Dozenten, Noten, und *Modulinstanzen* zugeordnet. Eine Modulinstanz ist eine konkrete Veranstaltung, die dem in der Prüfungsordnung vorgeschriebenen Modul zugeordnet ist. Als solche ist ihr ein Semester, ein (Unter-) Titel und eine eigene Anzahl von ECTS-Punkten zugeordnet. Eigene Leistungspunkte sind deswegen notwendig, weil die Instanz unter Umständen nur einen Teil eines Moduls bildet.

Studierende Neben Personenbezogenen Daten, die aus der entsprechenden Datenbank des Studierendensekretariats stammen, werden zu jeder/jedem Studierenden insbesondere deren/-dessen Prüfungen gespeichert. Diese bestehen im Wesentlichen aus einer Liste von Modulprüfungen (s.o.).

Auf diesen Datenstrukturen sind im Prototyp eine Sammlung von Funktionen enthalten. Diese Funktionen sind allesamt in der Spezifikationssprache selbst definiert. Dazu gehören:

Funktionen auf Grunddatentypen Auf dem Typ (zeitliches) Datum sind grundlegende Funktionen für den Vergleich von Daten (früher/später sowie die Überprüfung von Zeiträumen) und der korrekten Addition von Tagen/Monaten/Jahren definiert. Außerdem sind Funktionen für die Umrechnung Datum-Semester definiert. Ein Datum kann kurz „11.11.2004“ oder lang „11. November 2004“ angezeigt werden.
Für Noten sind Rundungs- und Mittelfunktionen (arithmetisches Mittel, gewichtetes Mittel) definiert, sowie verschiedene Anzeigemodi (z.B. „Seht gut (-)“ oder „1.3”) definiert.
Für ECTS-Punkte liegen die wesentlichen arithmetische Operationen vor.

Funktionen auf Studierenden, Studiengängen, Modulprüfungen Die Frage, welche Note eine gegebene Prüfung darstellt, ist bereits abhängig von der Prüfungsordnung, insbesondere davon, ob zusätzliche Prüfungen zur Notenverbesserung erlaubt sind oder nicht. Deswegen verfügen alle Funktionen für Modulprüfungen über einen Parameter, mit dem man eine Funktion zur Notenberechnung angibt. Zur Verfügung stehen Funktionen zur Berechnung der Abdeckung, d.h. des erfolgreich bestandenen Anteils des Studiengangs in verschiedenen Versionen. Prüfungen können auch nach Zeiträumen gefiltert werden, um z.B. herauszufinden, welche Leistungen innerhalb der Regelstudienzeit erbracht wurden.

Dokumente und Webseiten Zusätzlich zu den Funktionen auf den eigentlichen Datentypen sind auch Möglichkeiten vorhanden, Dokumente zu formulieren und mit den entsprechenden Daten zu füllen. Dazu wird zunächst ein allgemeines Dokumentformat definiert und dieses dann nach Bedarf in LaTeX, PDF, HTML oder einfach nur reinen Text umgewandelt. Mit Hilfe dieser Funktionalität erfolgt die Einbindung in das weiter unten beschriebene Web-Interface.

3.2 Spezifikation der Prüfungsordnung für den Bachelor-Studiengang Informatik

Der Prototyp umfasst die Spezifikation von zwei Varianten der Prüfungsordnung für den Studiengang „Bachelor Informatik“ an der Christian-Albrechts-Universität zu Kiel. Teile der Spezifikationen sind:

Konstanten wie z.B. Bezeichnung des Abschlusses, Anzahl der Semester für die Regelstudienzeit, Name der Universität und des Instituts, der Gesamtumfang an ECTS-Punkten etc.

Der Studiengang besteht aus einer Liste von Modulen zusammen mit einer Code-Bezeichnung. Teil der Spezifikation des Studiengangs ist die auf Seite 12 angegebene Menge der Modultypen und einfache Funktionen wie `istModulTyp`.

Abbildung einzelner Definitionen der Prüfungsordnung
Eine Prüfungsordnung enthält eine Vielzahl von logischen Zusammenhängen, Fristen und Begriffsdefinitionen. Für den Prototyp haben wir uns bemüht, diese vollständig abzubilden. Dazu gehören unter anderem Funktionen für die Berechnung der Note und Filter für die bestandenen Prüfungen. Für letztere gibt es auch zusätzlich die Möglichkeit einen Zeitraum anzugeben, also z.B. die bestandenen Prüfungen der letzten zwei Semester zu ermitteln. Die Berechnung der Abschlussnote erfolgt durch eine Mittlung anhand der ECTS-Punkte. Entsprechend der Vorschrift der Prüfungsordnung wird korrekt gerundet, und, falls entsprechende Daten vorliegen, der Antrag der/des Studierenden beachtet, einige Modulnoten bei der Berechnung der Endnote nicht zu berücksichtigen.
Eine Reihe von Funktionen befassen sich mit dem Projektmodul (der Abschlussarbeit):

- Es gibt eine Prüfung, ob ein Student zum Projektmodul zugelassen ist.
- Es kann auch berechnet werden, welche Module noch zur Zulassung fehlen.
- Falls ein gegebener Student bereits ein Projekt begonnen hat, kann dies mit einer Funktion zurückgeliefert werden und entsprechend das Datum des Beginns oder das Thema ermittelt werden.

Die Prüfungsordnung gibt auch eine Empfehlung aus, dass eine/ein Studierender, wenn sie/er unterhalb einer bestimmten Leistung bleibt, ein Beratungsgespräch suchen sollte. Auch diese Regelung wurde in einen entsprechenden Test umgesetzt.

Verschiedene Dokumente In der Prüfungsordnung sind außerdem Muster bzw. Beschreibungen für einige Dokumente enthalten, die das Prüfungsamt (evtl. auf Antrag) anzufertigen hat. Eine automatische Erstellung liegt vor für

- die Urkunde
- das Zeugnis
- Diploma Supplement
- Transcript of Records
- Bescheinigung erbrachter Leistungen (bei Exmatrikulation)

3.3 Import und Export per XML

Das implementierte Prüfungsverwaltungssystem sollte mit anderen an der Christian-Albrechts-Universität zu Kiel existierenden Applikationen zusammenarbeiten. Zu diesem Zweck waren Definitionen für die Umwandlungen von und nach XML[1] erforderlich. Aufgrund der hervorragenden Möglichkeiten zur Metaprogrammierung konnten Funktionen für diese Umwandlung automatisch aus der Spezifikation erzeugt werden. Für die Kooperation mit der Studierendendatenbank, die in Zukunft am Institut für Informatik der Christian-Albrechts-Universität zu Kiel für die Erfassung von Prüfungsleistungen eingesetzt werden wird, war deswegen lediglich eine einfache Transformation des von dieser Datenbank exportierten Formats erforderlich. Solche Transformationen sind grundsätzlich mit der Verwendung von XML verbunden. Um zumindest einen kleinen Einblick zu geben, wie man sich eine solche Umwandlung vorstellen kann, folgt ein kleines Beispiel:

In dem von der Datenbank exportierten Format ist jede einzelne Prüfung mit den Daten des Prüflings und des zugeordneten Moduls versehen. Hier ein Auszug aus den Daten für unseren Teststudent, der die Prüfung für ein Modul erst im zweiten Anlauf bestanden hat:

```
<result>
   <entry>
       <vorname>Julius</vorname>
       <nachname>Musterstudent</nachname>
       <geboren>1980-02-12</geboren>
       <studienbeginn>2001-10-01</studienbeginn>
       <matrnr>123456</matrnr>
       <titel>Organisation und Architektur von Rechnern
              </titel>
       <ectspunkte>7.0</ectspunkte>
       ...
       <note>5.0</note>
   </entry>
   <entry>
       <vorname>Julius</vorname>
       <nachname>Musterstudent</nachname>
       <geboren>1980-02-12</geboren>
       <studienbeginn>2001-10-01</studienbeginn>
       <matrnr>123456</matrnr>
       <titel>Organisation und Architektur von Rechnern
              </titel>
       <ectspunkte>7.0</ectspunkte>
       ...
       <note>2.3</note>
   </entry>
  ...
</result>
```

Unser Format sieht hingegen - neben anderen Unterschieden, auf die hier nicht eingegangen wird - eine stärkere Hierarchisierung der Daten vor:

[1] XML (eXtensible Markup Language) ist ein Standard zur Erstellung strukturierter, maschinen- und menschenlesbarer Dateien. XML definiert dabei den grundsätzlichen Aufbau solcher Dateien. Für die konkreten Anwendungsfälle müssen die Details des Dateiaufbaus jedoch weiter spezifiziert werden.

```
<student>
  <vorname>Julius</vorname>
  <nachname>Musterstudent</nachname>
  <geboren>1980-02-12</geboren>
  <studienbeginn>2001-10-01</studienbeginn>
  <matrnr>123456</matrnr>
  <[list]>
    <modulpruefung>
      <modul>
        <titel>Organisation und Architektur von Rechnern
               </titel>
        <ectspunkte>7.0</ectspunkte>
        ...
      </modul>
      <[list]>
        <pruefung>
          ...
          <note>5.0</note>
        </pruefung>
        <pruefung>
          ...
          <note>2.3</note>
        </pruefung>
      </[list]>
    </modulpruefung>
    ...
  </[list]>
</student>
```

Ein Teil der Transformation aus dem ersten Format in das unseres Prototyps sieht wie folgt aus:

```
transformStudent (XElem "result" (pruefung:pruefungen)) =
  personenDaten pruefung
                (map ohnePersonenDaten (pruefung:pruefungen))

ohnePersonenDaten (_:_:_:_:_:pruefung) = pruefung

personenDaten (vorname:nachname:geboren:beginn:nr:_)
              pruefungen =
  XElem "student"
    (vorname:nachname:geboren:beginn:nr:
     XElem "[list]" (map transformPruefung pruefungen))
```

Erklärung: Die Funktion `transformStudent` verarbeitet einen XML-Term mit der Bezeichnung `"result"`. Die darunterliegenden XML-Terme werden dabei in den ersten Term „`pruefung`“ und die restlichen Terme „`pruefungen`“ unterteilt. Mit der Funktion `personenDaten` werden die Angaben zur Person (die ersten fünf Terme) vorangestellt, während die ersten fünf Terme mittels `ohnePersonenDaten` entfernt werden. Terme, die nicht weiter verarbeitet werden, sind jeweils durch „_“ gekennzeichnet. Die zweimal auftretende Funktion `map` wendet eine gegebene Funktion auf jedes Element einer Liste an. Der erste Aufruf von `map` sorgt also dafür, dass die Personendaten für jeden einzelnen Term der Liste entfernt werden, während

der zweite Aufruf die restliche Transformation `transformPruefung` einleitet. Diese weitere Transformation wird hier nicht weiter ausgeführt, folgt aber den angegebenen Schemata.

3.4 Anbindung an bestehende Systeme

Mit Hilfe der im vorangegangenen Abschnitt beschriebenen Möglichkeiten zum Im- und Export konnten wir unseren Prototyp an bestehende Systeme anbinden, die in der Verwaltung unseres Instituts eingesetzt werden. Insgesamt können auf diese Weise fünf Systeme Daten miteinander austauschen, wie Abbildung 3.1 zeigt.

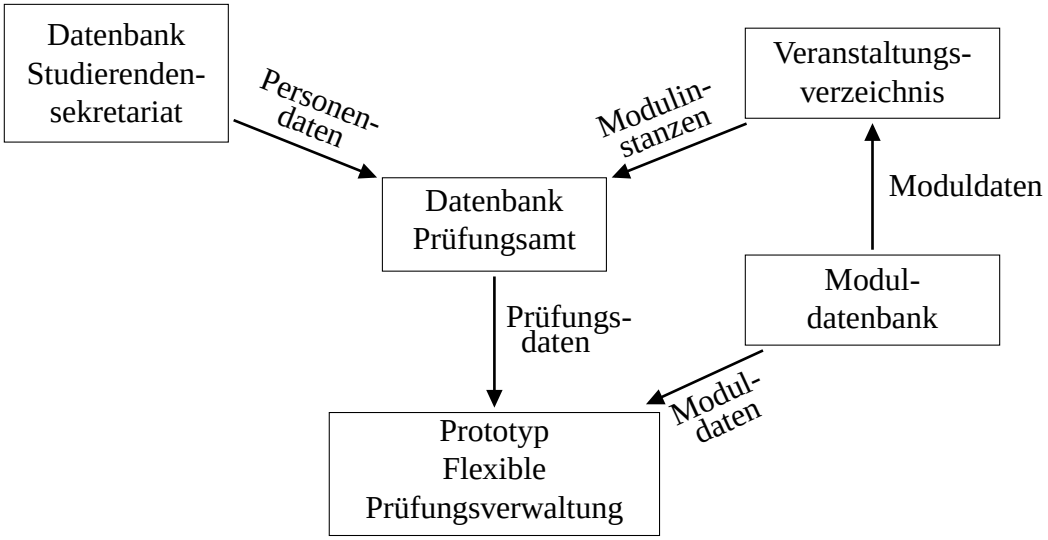

Abbildung 3.1: Datenaustausch zwischen verschiedenen Systemen

Die Daten zu jeder/jedem Studierenden wie Name, Matrikelnummer, Adresse werden im Studierendensekretariat aufgenommen und verwaltet. Außerdem existiert ein Veranstaltungsverzeichnis „UnivIS“, in dem zu jeder Lehrveranstaltung verantwortliche Dozenten, Räume, Termine, Angaben zum Inhalt und die entsprechenden Modulbezeichnungen abgelegt werden. Für die Zukunft geplant ist, das UnivIS mit der ebenfalls im Rahmen des BLK-Projektes erarbeiteten Moduldatenbank [2] zu verbinden, sodass die verwalteten Lehrveranstaltungen immer mit den richtigen Moduldaten versehen werden können. Teile der Information aus der Sekretariatsapplikation und dem UnivIS werden in der Datenbank des Prüfungsamtes benötigt und bei Bedarf angefordert. Der von uns erstellte Prototyp importiert die vom Prüfungsamt verwalteten Daten (die Ist-Daten in der Begrifflichkeit von Kapitel 2) und stellt anhand der Spezifikation der Prüfungsordnung zusätzliche Funktionalität (und damit die Herleitung der Soll-Daten) zur Verfügung. Sobald die Anbindung zwischen UnivIS und Moduldatenbank erfolgt ist, wird auch das flexible Prüfungsverwaltungssystem die erhaltenen Daten mit der Moduldatenbank abgleichen.

Eine solche Kooperation verschiedener Systeme ist aus mehreren Gründen gerade für Verwaltungsaufgaben sinnvoll: Die verwalteten Daten können zentral gehalten und aktualisiert werden. Hält hingegen jedes System seine eigene Kopie, so führt diese Redundanz unweigerlich zu Inkonsistenzen. Ändert sich zum Beispiel die Anschrift einer/eines Studierenden, so ist nicht zu erwarten, dass er seinen Umzug allen betroffenen Stellen mitteilt. Gibt es hingegen eine verantwortliche Stelle, die derartige Daten hält, so kann diese den anderen Systemen jederzeit die aktuellsten Daten mitteilen. Allein die Eingabe derselben Daten in verschiedene Systeme ist eine wesentliche Fehlerquelle. Denn bereits leichte Variationen oder Tippfehler in den Bezeichnungen führen dazu, dass Datensätze für dasselbe Objekt nicht mehr identifiziert werden können.

Voraussetzung für die Kooperation mehrerer Systeme ist aber, wie bereits ausgeführt, die Verwendung offener Austauschformate. Dies wird von herkömmlichen Systemen leider häufig noch nicht ausreichend beachtet.

3.5 Internetschnittstelle

Die in dieser Publikation beschriebene Realisierung des flexiblen Prüfungsverwaltungssystems wird als Erweiterung der im Prüfungsamt des Instituts für Informatik und Praktische Mathematik eingesetzten Studierendendatenbank verwendet. Diese Studierendendatenbank wurde mit konventionellen Mitteln erstellt[2] und die Erweiterung auf die von uns hinzugefügte Funktionalität wäre mit einem wesentlich höheren Aufwand verbunden. Die einzelnen Erweiterungen werden in den Abschnitten 3.6 und 3.7 vorgestellt.

Die Benutzer können das flexible Prüfungsverwaltungssystem über einen normalen Internetbrowser erreichen. Wenn die Webadresse des Systems aufgerufen wird, startet ein CGI-Skript. [3] Diesem Skript können weitere Spezifikationen (Matrikelnummer der/des Studierenden und ggf. der Name eines bestimmten Dienstes) als Parameter übergeben werden. Das Skript wurde mit Hilfe der deklarativen Programmiersprache `Curry` erstellt.[4] Wird dieses Skript aufgerufen, so erscheint eine Seite, auf der man die Matrikelnummer der/des Studierenden eingeben kann, für dessen/deren Daten man sich interessiert. Das Skript führt daraufhin eine SQL-Anfrage [5] bei der Studierendendatenbank durch, bei der auch überprüft wird, ob eine ausreichende Berechtigung vorliegt, um diese Daten einzusehen. Liegt die Berechtigung nicht vor, so wird der Benutzer automatisch auf den Login-Bildschirm der Studierendendatenbank geleitet.

Liegt die Berechtigung vor, werden die entsprechenden Daten im XML-Format übermittelt, siehe Abschnitt 3.3. Nach einer erfolgreichen Datenübermittlung wird dem Benutzer eine erste Übersichtsseite angezeigt, siehe Abbildung 3.2.

Für diese Übersichtsseite wurden bereits einige der in der Prüfungsordnung beschriebenen Tests durchgeführt. Z.B. kann man erkennen, dass die angezeigte Studentin aufgrund unter der Norm liegender Leistungen ein Beratungsgespräch suchen sollte. Per Knopfdruck können für den angezeigten Datensatz verschiedene Dienste aufgerufen werden. Diese Dienste werden näher in den Abschnitten 3.6 und 3.7 beschrieben. Die Ergebnisse aller Anfragen werden im HTML-Format dargestellt. Zusätzlich stehen dem Benutzer Dokumente und einige Informationen (z.B. die Liste der bisher erbrachten Leistungen) zum Download im PDF-Format zur Verfügung.

Der Datenaustausch zwischen der Studierendendatenbank und dem flexiblen Prüfungsverwaltungssystem erfolgt über das HTTP (HTTPS)-Protokoll [6]. Der Zugriff auf das flexibles Prü-

[2] Die Studierendendatenbank ist mit Hilfe der Programmiersprache `Java` entwickelt worden. Als Internetschnittstelle wird „`Tomcat`“, eine Erweiterung des Webservers zur Implementierung der Servlet-Spezifikation von Sun Microsystems, verwendet. Für die permanente Datenspeicherung wird die relationale Datenbank „`Ingres`“ verwendet.

[3] Über die CGI-Schnittstelle (Common Gateway Interface) können Programme im Internet bereitgestellt werden, die als Links von HTML-Dateien aus aufgerufen werden und die selbst HTML-Code erzeugen und diesen an einen Web-Browser senden. CGI stellt also eine Möglichkeit dar, Internetseiten dynamisch zu erzeugen.

[4] Weitere Informationen zur Programmiersprache `Curry` sind unter dem URL `http://www.informatik.uni-kiel.de/~curry` zu finden. PAKCS, die für das flexibles Prüfungsverwaltungssystem verwendete Implementierung von `Curry`, ist unter dem URL `http://www.informatik.uni-kiel.de/~pakcs` zu finden.

[5] SQL (Structured Query Language) ist eine Abfragesprache für relationale Datenbanken. Zur Erzeugung einer SQL-Anfrage verfügt die Realisierung des flexiblen Prüfungsverwaltungssystem über einen eigens dafür entwickelten Datentyp und entsprechende Funktionen.

[6] HTTP (Hypertext Transfer Protocol) ist ein zustandsloses Protokoll zur Übertragung von Daten über das Internet. HTTPS (Hypertext Transfer Protocol Secure) ist ein Netzwerkprotokoll, das eine gesicherte HTTP-Verbindung zwischen Rechnern ermöglicht. Dabei werden die Daten verschlüsselt, damit sie abhörsicher sind.

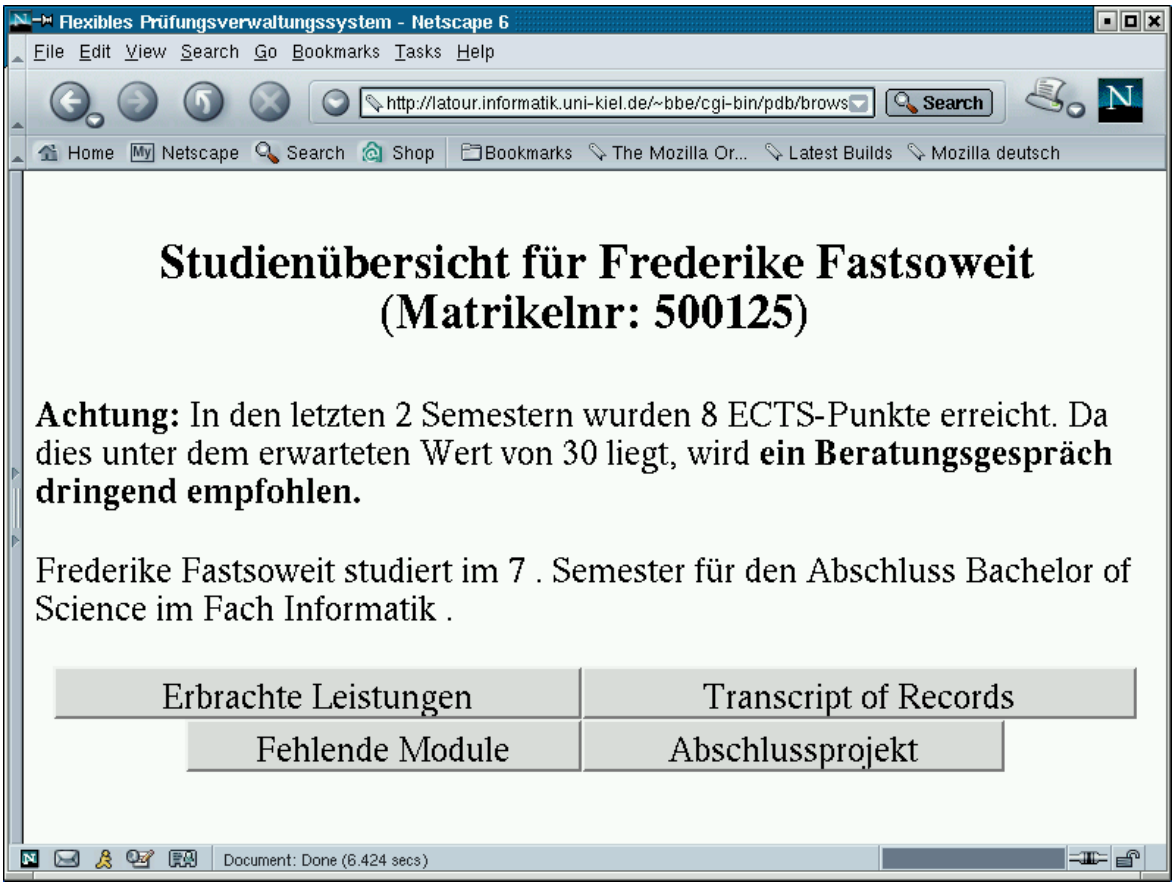

Abbildung 3.2: Übersichtsseite nach Auswahl eines Datensatzes

fungsverwaltungssystem ist nur autorisierten Benutzern möglich. Eine Benutzerverwaltung und einen entsprechenden Zugang regelt die Studierendendatenbank. Das flexibles Prüfungsverwaltungssystem sieht zur Zeit den Zugriff für die Mitarbeiter des Prüfungsamtes und für die Studierenden vor. Der ersten Benutzergruppe steht die volle Funktionalität zur Verfügung, die Studierenden haben dagegen nicht die Möglichkeit, Dokumente zu erstellen und können nur Einblick in ihre persönlichen Daten nehmen.

3.6 Automatische Dokumentenerstellung

Wie im Abschnitt 3.1 beschrieben, verfügt das flexible Prüfungsverwaltungssystem über ein allgemeines Format zur Definition von Dokumenten. Innerhalb des Systems spezifizierte Dokumente können je nach Bedarf in andere Formate (z.B. HTML, LaTeX oder PDF) umgewandelt werden. Dem Benutzer stehen Funktionen zur Erstellung der folgenden Dokumente zur Verfügung:

Urkunde Urkunde über die Erlangung des Bachelor-Abschlusses im Fach Informatik.

Zeugnis Das Zeugnis der Bachelor-Prüfung. Zeugnis und Urkunde entsprechen den in der Prüfungsordnung vorgegebenen Mustern.

Diploma Supplement Ein Diploma Supplement ist eine Anlage zum Zeugnis aus dem nähere Informationen zum Bachelor-Studiengang sowie die Einordnung des Bachelor-Abschlusses in Bezug auf internationale Maßstäbe hervorgeht.

Transcript of Records Eine Datenabschrift (Transcript of Records) wird der/dem Studierenden beim Hochschulwechsel ausgestellt.

Bescheinigung erbrachter Leistungen Eine offizielle Bescheinigung über erbrachte Leistungen wird der/dem Studierenden auf Antrag bei der Exmatrikulation ausgestellt, wenn der entsprechende Abschluss nicht erlangt werden konnte. Eine ähnliche Auflistung kann der Student jederzeit über die Internetschnittstelle abrufen, um sich einen Überblick über seine erbrachten bzw. noch zu erbringenden Leistungen zu informieren.

3.7 Dienste zur Abwicklung von Standardanfragen

Die in dem flexiblen Prüfungsverwaltungssystem realisierten Funktionen bzw. die von der Internetschnittstelle zur Verfügung gestellten Dienste entspringen der Prüfungsordnung und der für das System durchgeführten Anforderungsanalyse. Der Prototyp des flexiblen Prüfungsverwaltungssystems erweitert die im Prüfungsamt bereits eingesetzte Studierendendatenbank um einige Zusatzfunktionen. Neben den im vorangegangenen Abschnitt dargestellten Diensten zur automatischen Erstellung von Dokumenten sind auch die folgenden Dienste über die Internetschnittstelle abrufbar:

Bestehen der Bachelor-Prüfung Diese Funktion liefert die Auskunft darüber, ob die/der Studierende die Bachelor-Prüfung bestanden hat. Sollte die Bachelor-Prüfung noch nicht bestanden sein, erfolgt eine Auskunft über die noch fehlenden Prüfungsleistungen.

Zulassung zum Abschlussprojekt Auf Seite 12 wurden als Beispiel die Voraussetzungen für die Zulassung zum Abschlussprojekt (Projektmodul) angeführt. Dieser Dienst überprüft zu einer/einem spezifizierten Studierenden, ob diese Voraussetzungen vorliegen. Wenn bereits eine Abschlussarbeit begonnen wurde, wird das zugehörige Thema angezeigt. Sind die Voraussetzungen noch nicht erfüllt, werden die noch fehlenden Leistungen aufgelistet.

Notwendigkeit eines Beratungsgesprächs Gemäß der Prüfungsordnung sollten Studierende, die in einem Studienjahr weniger als 30 Leistungspunkte erwerben, ein Beratungsgespräch suchen. Diese Funktion überprüft die entsprechenden Leistungen und gibt bei Bedarf einen entsprechenden Hinweis auf der Übersichtsseite aus, siehe Abbildung 3.2.

Berechnung der Abschlussnote Bei der Berechnung der Abschlussnote sind mehrere Aspekte zu berücksichtigen. Die Gesamtnote der Bachelorprüfung berechnet sich aus dem Durchschnitt der Noten von Modulen im Gesamtumfang von 180 Leistungspunkten. Bei der Berechnung sind die Noten aller prüfungsrelevanten Module (Grund-, Aufbau-, und Wahlpflichtmodule sowie das Projektmodul) einzubeziehen. Die Noten der Module werden mit ihren Leistungspunkten gewichtet, wobei als Gewicht der Grundmodule die halbe Leistungspunktzahl anzusetzen ist. Werden alle Modulprüfungen innerhalb der Regelstudienzeit abgelegt, so werden auf Antrag der/des Studierenden die Prüfungsnoten von Modulen im Gesamtumfang von bis zu 18 Leistungspunkten nicht zur Bildung der Gesamtnote herangezogen. Dieser Dienst berechnet die Abschlussnote nach diesen Kriterien.

Resümee

Ein Teilprojekt im Rahmen des BLK-Verbundprojektes 2 war es, Konzepte für eine bessere EDV-Unterstützung der Hochschulverwaltungen vorzulegen. Diese Konzepte sollten dem erhöhten Aufwand bei der Erfassung, Verwaltung und Auswertung der in modularisierten Studiengängen anfallenden Studierenden- und Prüfungsdaten Rechnung tragen. Ein Ergebnis ist das vorliegende Konzept eines flexiblen Prüfungsverwaltungssystems.

Kern des Konzepts ist eine Beschreibungssprache für die Spezifikation von Prüfungsordnungen. Diese Beschreibungssprache ist zum einen *universell*, sie legt dem Benutzer daher keinerlei Beschränkung dahingehend auf, welche logischen Zusammenhänge er ausdrücken kann. Zum anderen ist die Beschreibungssprache nicht nur durch das System automatisch verarbeitbar sondern auch für den menschlichen Benutzer lesbar. Der Einsatz dieser Beschreibungssprache bringt die folgenden wesentlichen Vorteile:

Flexible Lösung individueller Probleme Der Einsatz der Beschreibungssprache erlaubt eine Flexibilität, die bisherige Systeme nicht bieten konnten, die aber in der gegenwärtigen Situation immer wichtiger wird: Die vom individuellen Benutzer benötigten Funktionen können realisiert werden, ohne dass er sich dabei auf eine vorgegebene Grundmenge vorgegebener Funktionen beschränken muss.

Zielorientierte Interpretation von Daten Herkömmliche Systeme verwalteten die eingegebenen Daten zuverlässig und erlauben einen schnellen Zugriff. Weil sich im flexiblen Prüfungsverwaltungssystem eine Spezifikation der gesamten Prüfungsordnung umsetzen lässt, ist es in der Lage, diese Daten auch auf die in der Ordnung formulierten Ziele hin zu interpretieren. So wird es möglich, nicht nur einen Überblick darüber zu bekommen, was der Studierende bereits geschafft hat sondern auch darüber, was noch zu leisten ist.
Ein zweiter wesentlicher Bestandteil des Konzepts ist die grundsätzliche Trennung von Ist- (tatsächlich abgelegte Prüfungen) und Soll-Daten (in der Prüfungsordnung formulierte Ausbildungsziele). Durch diese Trennung ergibt sich als weiterer wesentlicher Vorteil:

Flexible Wahl der Prüfungsordnung Dieselben erfassten Prüfungsdaten lassen sich in Bezug auf verschiedene Prüfungsordnungen interpretieren. So kann nützliche Information für Fragen geliefert werden wie „Was bringt der Wechsel auf die neue Studienordnung?“ Eine brauchbare Auskunft bei solchen Fragestellungen war bisher nur in einem ausführlichen Gespräch mit einem Berater möglich, der beide Ordnungen bis ins Detail durchdrungen hat.
Schließlich ist als wesentliches Teilkonzept die Unterstützung offener Austauschformate zu nennen. Der wesentliche Vorteil dabei ist:

Flexible Anbindung an bestehende Systeme
Damit bietet dieses Konzept gegenüber den bisher eingesetzten Realisierungen zur Verwaltung von Studierenden- und Prüfungsdaten entscheidende Vorteile. Seine prototypische Umsetzung und Anbindung an die am Institut für Informatik und Praktische Mathematik der Christian-Albrechts-Universität zu Kiel vorhandenen Softwarelösungen zeugt von seiner praktischen Einsetzbarkeit.

Literaturverzeichnis

[1] *BLK-Modellversuchsprogramm "Entwicklung eines Leistungspunktesystems an Hochschulen. Abschlussbericht. Verbund 2 "Entwicklung eines Leistungspunktesystems in Fachbereichen Elektrotechnik und Informatik"*. Hochschule Bremen, Bremen, 2004.

[2] Barbara Bennemann and Thomas Scheidsteger. *Konzept und prototypische Umsetzung einer verteilten heterogenen Moduldatenbank.* Carl von Ossietzky Universität Oldenburg, Oldenburg, 2004.

[3] Bundesministerium für Bildung und Forschung. *Bologna-Prozess. Hin zu einem Europäischen Hochschulraum bis 2010. Von Bologna nach Berlin.* BMBF, 2003.

[4] Kultusministerkonferenz. *Strukturvorgaben für die Einführung von Bachelor-/Bakkalaureus- und Master-/Magisterstudiengängen*. Beschluss der Kultusministerkonferenz vom 05.03.1999 in der Fassung vom 14.12.1999, Bonn, 1999.

[5] Kultusministerkonferenz. *Rahmenvorgaben für die Einführung von Leistungspunktsystemen und die Modularisierung von Studiengängen.* Beschluss der Kultusministerkonferenz vom 15.09.2000, Bonn, 2000.