Ich danke meinen Kollegen und Vorgesetzten in der Arbeitsgruppe Bibliotheksautomation für ihre freundliche Unterstützung meines Studiums; im besonderen Herrn DI Kampl für die Oracle-Manuals und Herrn Scherbaum für die Mac-Screenshots.

Außerdem danke ich allen, die mir geholfen haben, diese lange, nicht immer besonders witzige Zeit des Studiums erfolgreich hinter mich zu bringen.

A

DECKBLATT

Diplomarbeit

„Entwicklung eines webbasierten Informationssystems für Universitätsinstitute“

zur Erlangung des akademischen Grades

Magister rerum socialium oeconomicarumque

(Mag. rer. soc. oec.)

Magister der Sozial- und Wirtschaftswissenschaften

Sozial- und Wirtschaftswissenschaftliche Fakultät

Universität Wien

Eingereicht von:

Horst M. Eidenberger

(Mat. Nr. 94 25 694)

Betreuer/Begutachter:

Univ. Doz. Dr. Christian Breiteneder

Wien, im Oktober 1997

B

DIPLOMARBEIT

Inhalt

1. Einführung

1.1 Aufgabenstellung

Ziel der vorliegenden Diplomarbeit war es, für das bereits bestehende Institutsinformationssystem (IIS) des Institutes für Angewandte Informatik und Informationssysteme (ANIIS) eine webbasierte Benutzeroberfläche entsprechend dem WWW-Konzept des Institutes ([21]) zu entwickeln. Dabei sollten die einzelnen Webseiten aus Scripts generiert werden, die ihre Daten der IIS-Datenbank entnehmen, um die Aktualität der dargebotenen Informationen zu gewährleisten.

Schematisch läßt sich die Aufgabenstellung folgendermaßen darstellen:

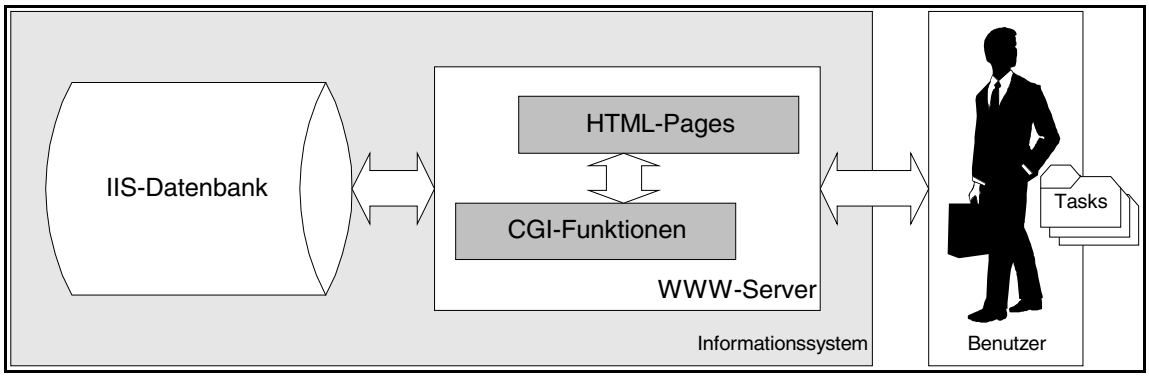

Abb. 1: Aufgabenstellung

Die Güte eines Webserver hängt wesentlich von der Aktualität der angebotenen Informationen ab. Zwar legt man heute auch großen Wert auf eine ansprechende (und ressourcenintensive) visuelle Präsentation der Daten, entscheidend für den Benutzer ist aber immer noch der Informationsgehalt. Die angebotenen Informationen aktuell zu halten, ist Aufgabe der für sie zuständigen Personen (so ist z. B. jeder Lektor für die von ihm angebotenen Lehrveranstaltungen verantwortlich), der Systemplanungsprozeß kann hier nur insofern unterstützend wirken, indem er einerseits klare Zuordnungen von Personen zu Datenobjekten schafft und andererseits – durch eine möglichst scharfe Trennung statischer von dynamischen Systemkomponenten – den Wartungsaufwand minimiert. So wird beispielsweise durch die Generierung von HTML-Seiten aus Vorlagen, die Datenbankabfragen anstelle von konkreten Datenwerten enthalten, erreicht, daß mehrere Views auf dieselben Daten möglich sind, ohne sie mehrfach warten zu müssen. Außerdem muß der Wartungsvorgang an sich nicht durch (unsystematisches) Editieren von Textdateien erfolgen. Es liegt nahe, dynamische Systemteile (z. B. Mitarbeiterinformationen) in Form von Datenbankabfragen und statische als Textvorlagen (z. B. im HTML-Format) zu implementieren.

Neben der Minimierung des Wartungsaufwandes war es ein wichtiges Ziel, das bisherige Datenbankschema möglichst weitgehend beizubehalten und - wo nötig - zu erweitern, sodaß die bisher verwendeten SQL-Forms-Anwendungen auch weiterhin benützt werden können. Ursprünglich war vorgesehen, die Datenbank zu

vereinheitlichen und, da sie in mehreren Stufen entwickelt wurde und einige Redundanzen enthält, zu normalisieren. Deshalb wurde zusätzlich zum erweiterten alten Schema ein neues Datenmodell entwickelt sowie die Voraussetzungen (Taskplan, Entwicklungsumgebung, usw.; s. u.) geschaffen, um die darauf basierenden Anwendungen (z. B. Projektverwaltung) schrittweise zu implementieren.

Ein weiteres wichtiges Kriterium war der Wunsch, die neue Weboberfläche schon im Wintersemester 1997/98 zur Lehrveranstaltungsanmeldung zu verwenden. Das WWW-Konzept des Institutes schätzt aber allein für die Neu-Implementierung einen Arbeitsaufwand von vier Mannjahren (26 Wochen mal vier Personen). Zwar handelt es sich bei der vorliegenden Arbeit um keine vollständige Neuentwicklung, aber um die obige Deadline einzuhalten, war es dennoch nötig, einerseits das Gewicht des Entwicklungsprozeßes von der Planung auf die Implementierung zu verlagern und andererseits den Programmieraufwand durch Vermeiden von Mehrfach-entwicklungen zu minimieren. Dazu wurde ein eigenes, vom Designansatz abgeleitetes Vorgehensmodell entwickelt.

Die Benutzeroberfläche sollte neben einer adäquaten Hauptseite als Einstiegspunkt ins IIS über folgende Komponenten verfügen:

1. Verwaltung von Lehrveranstaltungen (LVA)

Studenten sollten die Möglichkeit haben, Informationen über aktuelle Lehrveranstaltungen zu lesen, sich zu Lehrveranstaltungen an- bzw. von LVAs abzumelden sowie ihren persönlichen LVA-Status (Anmeldestatus, Beurteilungen) zu lesen.

2. Verwaltung von Prüfungen

Auch her waren die wesentlichen Funktionen: Informationen lesen, An- und Abmeldung, sowie den persönlichen Prüfungsstatus lesen.

3. Whiteboard-System

Im IIS eingetragene Personen (Lektoren und Studenten) sollten die Möglichkeit haben, nach Kategorien geordnete Mitteilungen ins Whiteboard (Metapher: Schaukasten) einzutragen und eigene Einträge zu bearbeiten bzw. zu löschen. Jeder Besucher der Institutshomepage sollte das Whiteboard – kategorienweise oder ganz – einsehen, bzw. nach Schlagworten durchsuchen können. Darüber hinaus war für IIS-Manager (Personen mit umfassenden Rechten in der Datenbank) eine Löschfunktion für unerwünschte Eintragungen vorzusehen.

4. Mitarbeiterinformationen

Es sollte möglich sein, aus den Informationen in der Datenbank eine Liste aller aktuellen Mitarbeiter des Institutes mit Links auf Seiten mit Detailinformationen (wie z. B. Sprechstunde, usw.) zu generieren.

5. Verwaltung von Publikationen

Schließlich war eine einfache Komponente zur Verwaltung und Präsentation von Publikationen vorzusehen.

1.2 Vorgehensmodell

Da es sich beim IIS um eine dezidiert benutzerorientierte Anwendung handelt, lag es auf der Hand, einen designorientierten Systemplanungsansatz zu wählen. Von den beiden bekannten, an der Uni-Wien gelehrten Prototyping-Ansätzen wurde jener von Redmond-Pyle und Moore gewählt, wie er in [5] und [15] beschrieben ist. Gegenüber seinem Konkurrenten, dem Usability-Engineering nach [16] bietet er folgende Vorteile:

- Die einzelnen Phasen des Entwicklungsprozeßes sind genauer abgegrenzt und detailliert. Der Anwender wird mehr durch den Planungsprozeß geführt als beim Usability-Engineering, wodurch das Verfahren leichter anwendbar ist.
- Es wird mehr Wert auf Modellierungsaspekte gelegt. Zum Verständnis der wesentlichen Aspekte und um Doppelgleisigkeiten in der Realisierung zu vermeiden, ist ein gutes Modell des Systems notwendig. Usability Engineering steuert zum Verstehen der Aufgabenstellung nicht viel mehr als "Know the user" bei.
- Die Reihenfolge der Phasen ist klarer vorgegeben als beim Usability-Engineering, aber weniger strikt als beim klassischen Wasserfallmodell; das Redmond/Moore-Verfahren bietet also einen attraktiven Kompromiß zwischen der Dynamik des ersteren und der Effizienz des letzteren.

Das Redmond/Moore-Vorgehensmodell läßt sich graphisch folgendermaßen darstellen (aus [5]):

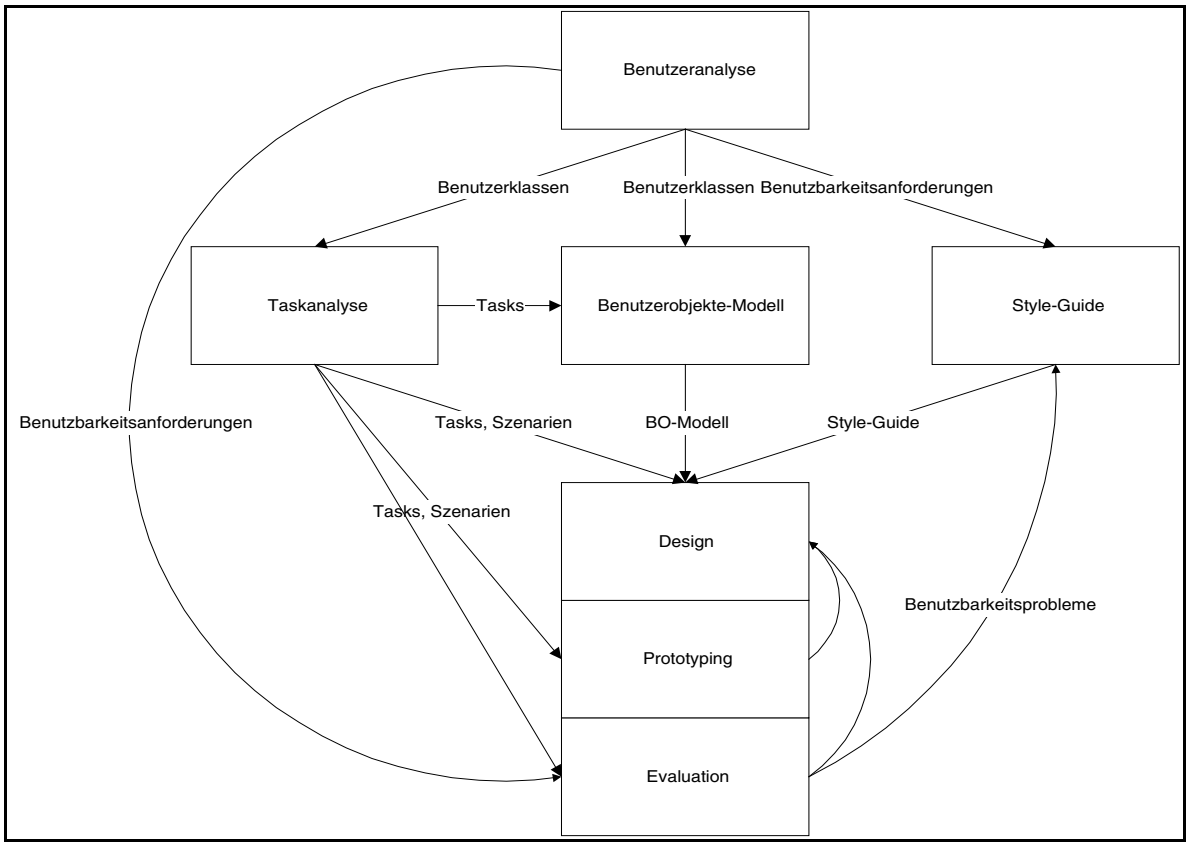

Abb. 2: Design-Vorgehensmodell

Das Verfahren beginnt mit dem Herausarbeiten der verschiedenen Benutzerklassen und ihren Anforderungsprofilen. Sodann werden in der Benutzeranalyse für jede Benutzergruppe die Benutzbarkeitsanforderungen für das System festgelegt. Es wird versucht, diese Kriterien (z. B. Performance, Flexibilität, Fehlertoleranz, usw.) möglichst konkret zu operationalisieren (z. B. kann eine Antwortzeit von weniger als drei Sekunden im Mittel über alle Testpersonen verlangt werden). Es ist allerdings fraglich, inwieweit eine so genaue Festlegung in Zahlen (in diesem frühen Stadium des Entwicklungsprozeßes) sinnvoll ist; möglicherweise ist es nur das kunstvolle Aufschreiben von wissenschaftlich generierten Hausnummern.

Auf die Benutzeranalyse folgt die Erarbeitung der Benutzeraufgaben (Taskanalyse). Ziel ist es, alle mithilfe des Systemes zu erledigenden Aufgaben zu erfassen, möglichst genau zu detaillieren und durch Szenarien verständlich und für die spätere Programmevaluation operationalisierbar zu machen. Kennt man die Benutzer und ihre Tasks, kann man daran gehen, die wesentlichen Benutzerobjekte herauszufiltern und zueinander in Beziehung zu bringen. Redmond/Moore haben dafür eine eigene Diagrammtechnik entwickelt, die man aber - unter geringem Informationsverlust - mit der EER-Diagrammtechnik substituieren kann. Außerdem wird in dieser Phase festgelegt, welche Systemaktionen (z. B. Ändern, Löschen, Drucken, usw.) auf welche Benutzerobjekte ausgeführt werden dürfen.

Anschließend definiert man die grundlegenden Konzepte für die Benutzerschnittstelle (Style Guide). Durch diese globalen Festlegungen (von Überschriften, Screen-Einteilung, Fehlermeldungen, usw.) wird ein konsistenter Look-and-Feel erreicht, wie er u. a. in der ISO-Norm 9241, Teil 10 verlangt wird.

Damit ist der Planungsprozeß abgeschlossen, und es beginnt - abgesehen von nachträglichen Änderungen - die Realisierung in Form von iterativem, zumeist evolutionärem Prototyping. Eine Prototypingphase besteht dabei stets aus dem Design der zu realisierenden Komponenten, der Entwicklung eines Prototypen und der anschließenden Evaluation mithilfe von Testpersonen, Experten oder Heuristiken. Aufgedeckte Benutzbarkeitsprobleme führen zu einem Redesign bzw. zur Adaptierung des Prototypen. Prototyping nennt man evolutionär, wenn entwickelte und getestete Komponenten in späteren Iterationszyklen wiederverwendet werden und schließlich in das fertige System miteinfließen.

1.3 Erweiterungen und Änderungen des Vorgehensmodelles

In der Systemplanung kann man leicht den Eindruck gewinnen, daß es eine Anzahl von Vorgehensmodellen gibt, von denen das passendste für die jeweilige Aufgabenstellung ausgewählt wird und sodann die fragliche Anwendung entsprechend diesem Modell entwickelt wird. Das ist, als ob man in der Realisierung zuerst das Betriebsystem festlegt und dann die Applikation dafür maßgeschneidert produziert: das Problem wird an die realen Gegebenheiten angepaßt und nicht umgekehrt, wie es sein sollte (vgl. [3], Kapitel 4.4).

Folgende Gründe sprechen für die Anpassung des Vorgehensmodelles an die IIS-Aufgabenstellung:

- Der Redmond/Moore-Ansatz bezieht sich hauptsächlich auf die Entwicklung der Benutzerschnittstelle, die eigentliche Anwendungsentwicklung wird nur, wo sie für das Funktionieren des Prototypen wichtig ist, berührt.
- Das Vorgehensmodell integriert keine Datenmodellierungsaspekte.
- Schließlich ist es phasenorientiert, d. h. es wird angewendet mit zwei, drei, usw. Prototypingphasen, die in sich geschlossen sind. Die Ergebnisse der ersten Phase sind für die zweite schon unveränderlich, usw.

Im Bereich der Webanwendungen bietet sich aber eine Alternative zum phasenorientierten Vorgehen, die ich kontinuierliches Prototyping nennen möchte. Auch hier erfolgt das Prototyping phasenorientiert; die Evaluation ist aber nicht mehr an die Prototypingphasen gekoppelt, sondern erfolgt nach der Freigabe eines Prototypen unabhängig, bis die untersuchte Komponente ein für die Benutzer befriedigendes Niveau erreicht hat. Die Vorteile dieser Sichtweise sind:

- Die Benutzer (Testpersonen) können direkt auf die Programmierung ihres Systemes einwirken. Nachdem der Systementwickler, um die große, alles enthaltende Leere zu überwinden, einen Lösungsvorschlag für ein Problem gemacht hat (Prototyp), können sie das System durch die Angabe von Änderungswünschen ihren Vorstellungen entsprechend gestalten. So wird das Grundübel, daß ein Systemplaner ein System entwickelt, während die betroffenen Benutzer nur zuschauen, weitgehend beseitigt.
- In der Folge sollten die Benutzer höher motiviert sein, da ihr Einfluß größer ist. Eventuell könnten aber auch weniger erfreuliche Motive für die höhere Motivation verantwortlich sein ...
- Schließlich sollten die Produkte dieses Entwicklungsprozeßes - da sie ja weitgehend den Benutzerwünschen entsprechen - besser sein.

Warum soll aber gerade die Webprogrammierung besonders gut für dieses Verfahren geeignet sein? Der Grund ist einfach, daß eine Webapplikation nur von Benutzern gewünscht wird, die selbst Zugang zum Medium haben und nur mithilfe des Internets entwickelt werden kann. Es besteht also zwangsläufig eine sehr gute Kommunikationsbasis zwischen den Entwicklungspartnern. Durch die Verwendung von eMail, d. h. der Kommunikation über festgelegte Adressen, wird darüber hinaus eine störungsfreie und sehr flexible Kommunikation garantiert. Schließlich ist bei Webprogrammen, da man die einzelnen Pages zumeist direkt vom Server lädt, garantiert, daß man stets mit der aktuellen Version arbeitet.

Alle diese Vorteile fehlen bei der herkömmlichen Entwicklung eines Clients auf einem lokalen System, was m. E. - neben der mangelnden Eignung von Fat Clients für prototypingorientiertes Vorgehen - ein entscheidendes Argument gegen Java ist. Allerdings wird die Systementscheidung erst bei der Modellierung (Kapitel 2.4) relevant.

Das erweiterte und angepaßte Vorgehensmodell hat das folgende Aussehen:

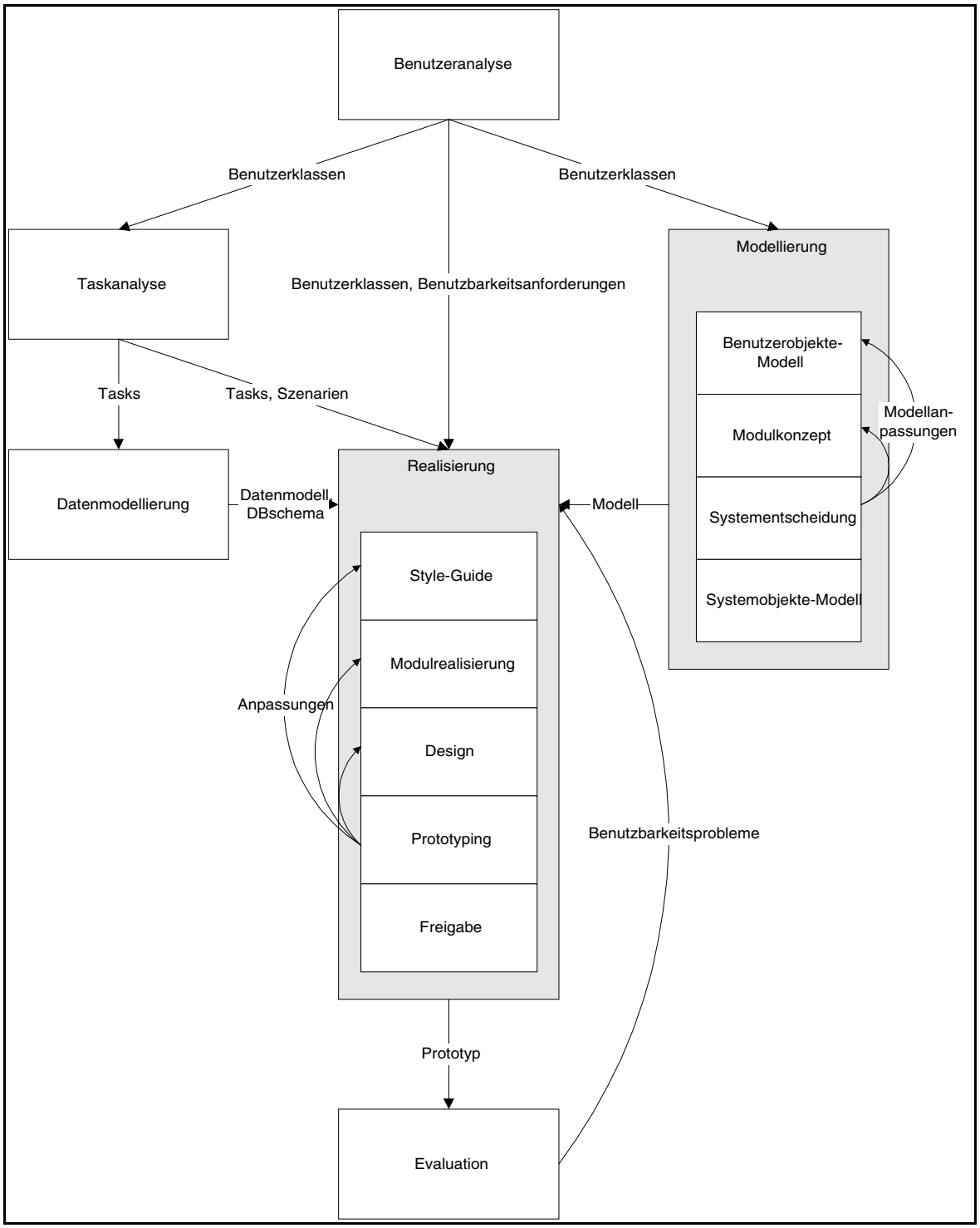

Abb. 3: Erweitertes Vorgehensmodell

Auf die Benutzeranalyse folgt weiterhin die Taskanalyse. Gleichzeitig kann jedoch die Modellierungsphase beginnen, in der zuerst das Benutzerobjekte-Modell erstellt wird. Danach wird in der Phase Modulkonzept versucht, möglichst viele Tasks und Subtasks zu vereinheitlichen und für die spätere Programmierung zu modellieren. Durch dieses bottom up-Vorgehen wird erreicht, daß Mehrfachentwicklungen möglichst vermieden werden. Das Hauptergebnis dieser Phase ist eine Bibliothek von Programmierbausteinen, aus denen sich die späteren Prototypen, bis hin zum fertigen System, zusammensetzen. Danach muß die Systementscheidung erfolgen. Es ist dabei von außen nach innen vorzugehen, d. h. erst ist die Entwicklungs-

umgebung festzulegen, sodann die Datenhaltungskomponente und erst am Schluß das Betriebsystem und die Hardwareumgebung. In Analogie zum Benutzerobjekte-Modell kann man dann noch zum besseren Verständnis des Systemes ein Systemobjekte-Modell erarbeiten, das die wesentlichen Aspekte der Problemstellung, wie sie sich zu diesem Zeitpunkt darstellen, zueinander in Beziehung bringt. Als Notation empfiehlt sich eine möglichst selbsterklärende Schreibweise, wie z. B. die ER-Notation.

Von der Modellierung unabhängig beginnt nach der Taskanalyse die Datenmodellierungsphase. Wird ein neues System geplant, so ist hier in der üblichen, u. a. in [2], [9] und [10] dargestellten Form vorzugehen; handelt es sich um die Erweiterung eines bereits bestehenden Systemes, empfiehlt es sich, erst das bestehende Datenmodell vernünftig zu dokumentieren, bevor man es erweitert. Im übrigen sollte man Änderungen bestehender Datenschemata möglichst vermeiden, da es normalerweise weniger Aufwand ist, die entsprechenden Komponenten neu zu entwickeln.

Die Realisierungsphase (oder: Kontinuierliches Prototyping) beginnt mit der Festlegung des Style-Guides, also dem Grobdesign der späteren Anwendung. Auf diese Phase folgt die Modulrealisierung, welche die in der Modulkonzept-Phase isolierten Systembausteine realisiert, austestet und in einer Bibliothek für das Prototyping zur Verfügung stellt. Anschließend werden, entsprechend einem Prototyping-Plan, Systemkomponenten - unter Bedachtnahme auf den Style-Guide und die Benutzbarkeitsanforderungen - designed und ein Prototyp erstellt. Der Prototyping-Plan gliedert die Liste aller zu realisierenden Tasks in Gruppen, die gemeinsam in einem Prototypingzyklus behandelt werden. Ist der Prototyp einer Gruppe von Funktionen fertig erstellt und ausgetestet, so wird er für die Evaluation freigegeben. Die Testuser können diesen Prototypen dann verwenden und durch ihr Feedback ihren Wünschen entsprechend umgestalten.

Da im Laufe dieses Systementwicklungsprozeßes der Aufwand für Änderungen immer mehr zunimmt, ist bei der Erarbeitung des Prototyping-Planes darauf zu achten, daß die Gruppengröße mit der Zeit abnimmt. Sind beispielsweise 14 Tasks zu realisieren und will man in drei Gruppen vorgehen, so empfiehlt sich eine Einteilung wie z. B.: 8/4/2 oder 7/5/1.

1.4 Projektorganisation

Da es sich um eine Einzelarbeit handelte, kam der Projektorganisation nicht die Bedeutung wie in einem Mehrpersonenprojekt zu. Es wurde linear vorgegangen:

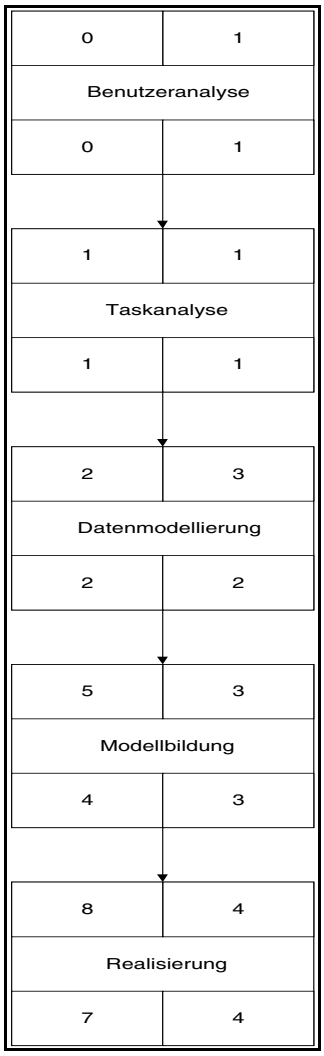

Abb. 4: Netzplan

Legende: Bei jeder Aktivität steht links oben der Sollbeginn, rechts oben die Solldauer, links unten der Istbeginn und rechts unten die Istdauer (Angaben in Wochen).

Das Projekt hatte also eine Gesamtdauer von zweieinhalb Monaten. Begonnen wurde mitte Juli. Das System war also Ende September - abgesehen von unwesentlichen Änderungen - fertiggestellt.

Als Ansprechpartner standen folgende Personen zur Verfügung:

Bereich	Name
Aufgabenstellung	Dr. Christian Breiteneder
Entwicklungsumgebung	Jan Stankowski
Datenbank	Michaela Pröll
Testpersonen	Dr. Breiteneder, Dr. Hitz, Dr. Polaschek, ...

Tab. 1: Ansprechpartner

Als Kommunikationsmedium wurde vor allem eMail eingesetzt, da es neben der Informations- noch Dokumentationsfunktion erfüllt. In periodischen Abständen (vor

Beginn / nach Ende einer Projektphase) wurden Besprechungen mit Dr. Breiteneder und betroffenen Dritten abgehalten.

Zur Implementierung des Systemes stand ein Sun-Rechner mit Solaris als Betriebsystem sowie einer Oracle 7 - Datenbank zur Verfügung.

1.5 Zum Aufbau der Arbeit

Die Arbeit gliedert sich in vier Abschnitte, die jeweils in mehrere Kapitel unterteilt sind: Deckblatt, Diplomarbeit, Verzeichnisse und Anhang. Im ersten Kapitel des zweiten Abschnittes wird die Aufgabenstellung erläutert, anschließend ein passendes Vorgehensmodell für die Systemplanung eingeführt, das dann an die aktuellen Gegebenheiten angepaßt wird, bevor Aspekte der Projektorganisation im Kapitel 1.4 zusammengefaßt werden.

Der zweite Teil des zweiten Abschnittes stellt die Ergebnisse des Entwicklungsprozeßes dar. Anschließend wird die Installation des Systemes geschildert. Kapitel 2.8 faßt den zweiten Teil mithilfe des Projekttagebuches zusammen. Der dritte Teil beschäftigt sich mit dem erstellten System; es werden wichtige verwendete Programmierkonzepte erläutert. Der vierte Teil beschäftigt sich mit der Entwicklungsumgebung, die für dieses System geschaffen wurde und die sich auch auf andere Aufgabenstellungen anwenden läßt.

Bevor im sechsten Teil nochmals eine Zusammenfassung des zweiten Abschnittes folgt, behandelt der fünfte Teil weiterführende Überlegungen, die u. a. das Datenmodell, die Entwicklungsumgebung, usw. betreffen. Schematisch läßt sich der zweite Teil folgendermaßen darstellen:

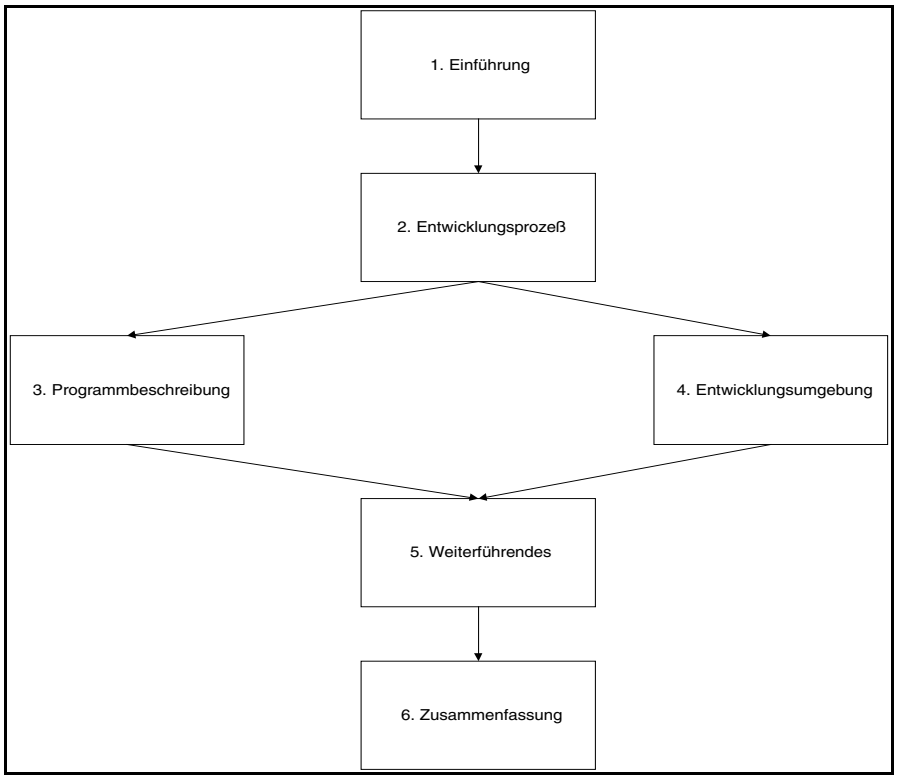

Abb. 5: Gliederung der Diplomarbeit

Der vierte Abschnitt enthält einen ausführlichen Anhang, in dem unter anderem Screenshots von Web-Pages und ein erweitertes Datenmodell für die Weiterentwicklung des IIS enthalten sind.

Die Testversion des IIS findet sich unter der URL:

http://joplin.pri.univie.ac.at/~eidenb/main.html

2. Entwicklungsprozeß / Ergebnisse

2.1 Benutzeranalyse

Im Rahmen der Benutzeranalyse wurden, unter Berücksichtigung der jeweiligen Zugriffsrechte, die verschiedenen Benutzergruppen und ihre Aufgabenprofile erhoben. Danach wurden die Benutzbarkeitsanforderungen an das System festgelegt.

Logisch können vier Benutzergruppen unterschieden werden:

#	Gruppe	Beschreibung
1	IIS-Manager	IIS-Manager sind v. a. für die Verwaltung der meisten Stammdaten (wie Studentenstammdaten, usw.) sowie für die Vergabe der Zugriffsrechte zuständig.
2	Lektoren	Lektoren verwalten ihre Lehrveranstaltungen und Prüfungstermine. Unter der Bezeichnung Lektoren sind Professoren, Dozenten, Assistenten und alle übrigen Universitätsmitarbeiter zusammengefaßt.
3	Studenten	Studenten melden sich zu Lehrveranstaltungen, Prüfungen an, wieder ab oder lesen ihren Prüfungsstatus. Diese Funktionen beziehen sich nur auf die im IIS eingetragenen Studenten.
4	Besucher	Alle übrigen Personen haben nur Lesezugriff auf Lehrveranstaltungsankündigungen, Lektoren-Homepages, usw.

Tab. 2: Benutzergruppen

Tatsächlich unterscheidet das alte IIS (von dem die Datenbankstruktur weitgehend übernommen wurde) auf Tabellenebene nur zwischen Lektoren und Studenten. Letztendlich sind für die Zugehörigkeit einer Person zu einer Benutzergruppe aber nur ihre Zugriffsrechte entscheidend. Das alte IIS basiert auf diskreter Sicherheitspolitik (siehe [11]), sodaß die Zugriffsrechte jeder Person für jede Aktion auf jede Tabelle genau spezifiziert werden können.

Für die obige, logische - und in der Folge als Gruppeneinteilung weiter verwendete - Gliederung lassen sich die Sicherheitsstufen folgendermaßen darstellen:

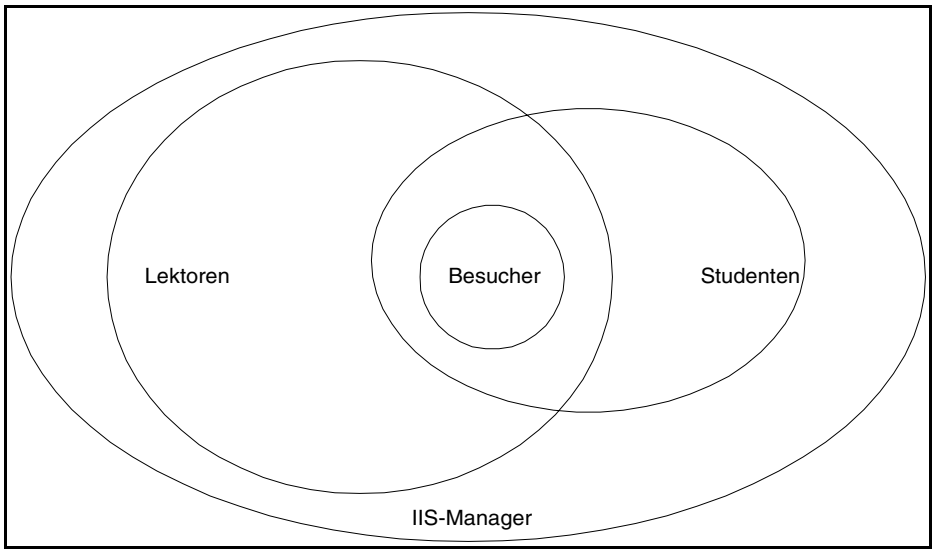

Abb. 6: Sicherheitslevel

Während IIS-Manager umfassende Rechte besitzen, haben Lektoren und Studenten eine - teilweise überlappende - Teilmenge dieser Rechte. In der Schnittmenge liegt zwangsläufig die Rechtemenge der Besucher.

Die Benutzbarkeitsanforderungen wurden in Form einer Heuristik formuliert:

#	Anforderung	Tool
1	Selbsterklärende Bildschirmmasken	Verwendung von Kommentaren
2	Direktes Feedback	Standardisierte Fehlermeldungen und Bestätigungen
3	Einheitliche Masken	Style-Guide, WWW-Konzept
4	Möglichst wenige Interaktionsstufen in Abläufen	Kompakte und vollständige Masken
5	Funktionale Vollständigkeit	Detaillierte Taskanalyse
6	Fehler-Hotline	Webmaster-Link in jeder Maske
7	Fehlertoleranz	Flexible Programmierumgebung
8	Einfache Erlernbarkeit	Style-Guide, Verwendung der Benutzersprache
9	Möglichst gute Performance	Effizienter Datenbankzugriff, Seitengenerierung
10	Gewährleistung des Datenschutzes	Persönliche Anmeldung vor dem Zugriff auf sensitive Daten

Tab. 3: Benutzbarkeitsanforderungen

Da der Schwerpunkt auf der Implementierung lag, wurde auf die Operationalisierung dieser Kriterien durch mittlere Antwortzeiten, Fragebögen, usw. verzichtet. Die Überprüfung der Benutzbarkeitsanforderungen erfolgte daher durch das selbständige Testen der Benutzerschnittstelle durch die Testuser und die Angabe von Änderungswünschen. Die Evaluation sollte ein "Try-to-destroy-it"-Wettbewerb sein.

2.2 Taskanalyse

Im Rahmen der Taskanalyse wurden für jede Benutzergruppe die mithilfe des Systemes auszuführenden Aufgaben festgestellt. Anschließend wurden sie nach ihrer Dringlichkeit zu Gruppen zusammengefaßt und in Subtasks zerlegt, die einen Hauptinput für die Modellbildung darstellten.

Task	Gruppe	Benutzer
Studentenlogin	1	S
LVA anmelden	1	S
LVA abmelden	1	S
LVA-Status lesen	1	S
PR anmelden	1	S
PR abmelden	1	S
PR-Status lesen	1	S
Logout	1	E
LVA-Informationen lesen	1	A
Prüfungsinformationen lesen	1	A
Lektorenlogin	2	L
Studentenstammdaten ändern	2	S
Studenten-Paßwort ändern	2	S
Whiteboard (WHB)-Eintrag hinzufügen	2	E
WHB-Eintrag ändern	2	E
WHB-Eintrag löschen	2	E
Whiteboard lesen	2	A
WHB-Kategorie hinzufügen	3	M
WHB-Kategorie ändern	3	M
WHB-Kategorien löschen	3	M
IIS-Manager WHB-Zensur	3	M
Abgelaufene Whiteboardeinträge löschen	3	BS
Mitarbeiterinformationen ausgeben	3	A
Publikationen hinzufügen	4	L
Publikationen ändern	4	L
Publikationen löschen	4	L
Publikationen ausgeben	4	A

Tab. 4: Taskplan

Benutzer: Studenten, Lektoren, IIS-Manager, Eingetragene Personen (L, M, S), Betriebsystem, Alle (Eingetragene und Besucher).

Da es nicht sinnvoll ist, hier alle Subtask-Zerlegungen anzuführen, wird als Beispiel nur die Zerlegung des Vorganges "Lehrveranstaltungen anmelden" wiedergegeben:

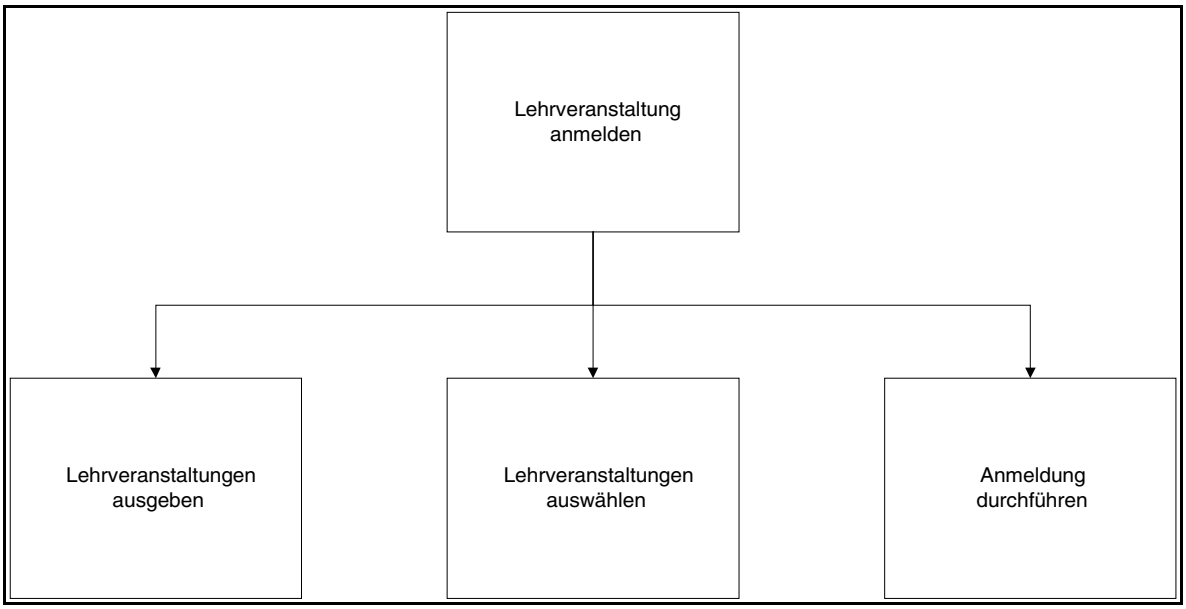

Abb. 7: Task-Zerlegung

Nach dem Auswählen der Option "Lehrveranstaltungen anmelden" werden alle aktuellen Veranstaltungen ausgegeben, der Benutzer kann dann seine Lehrveranstaltungen auswählen und durch einen Link die Anmeldung durchführen.

Um das Verständnis für die Problemstellung zu verbessern und als Voraussetzung für benutzerorientierte Testmethoden, werden im Rahmen der Taskanalyse auch Szenarien für die Erledigung von Aufgaben gebildet. Ein Szenario für die Aufgabe "Lehrveranstaltungen anmelden" könnte so aussehen:

#	Vorgang
1	Student wählt die Institutshomepage an
2	Mithilfe von "LVA-Informationen lesen" informiert er sich über das Lehrveranstaltungsangebot
3	Mit seinem Paßwort und seiner Matrikelnummer loggt er sich im Anmeldesystem ein
4	Nach Anwählen von "LVAs anmelden" wird eine Liste mit den aktuellen Lehrveranstaltungen (Titel, Nummer, usw.) ausgegeben.
5	Durch Ankreuzen der jedem Listeneintrag vorgestellten Checkbox markiert er seine LVAs
6	Durch das Anclicken des Submit-Buttons am unteren Ende der Anmeldemaske führt er die Anmeldung durch.
7	Das System bestätigt die erfolgreiche Anmeldung.

Tab. 5: Task-Szenario

2.3 Datenanalyse

Ziel der Datenanalyse war es, ausgehend vom Datenbankschema des alten IIS das Datenmodell zu erarbeiten und es anschließend um die für die zusätzlichen Systemfunktionen notwendigen Tabellen zu erweitern.

Das Datenmodell der alten IIS-Datenbank läßt sich als ER-Diagramm folgendermaßen darstellen:

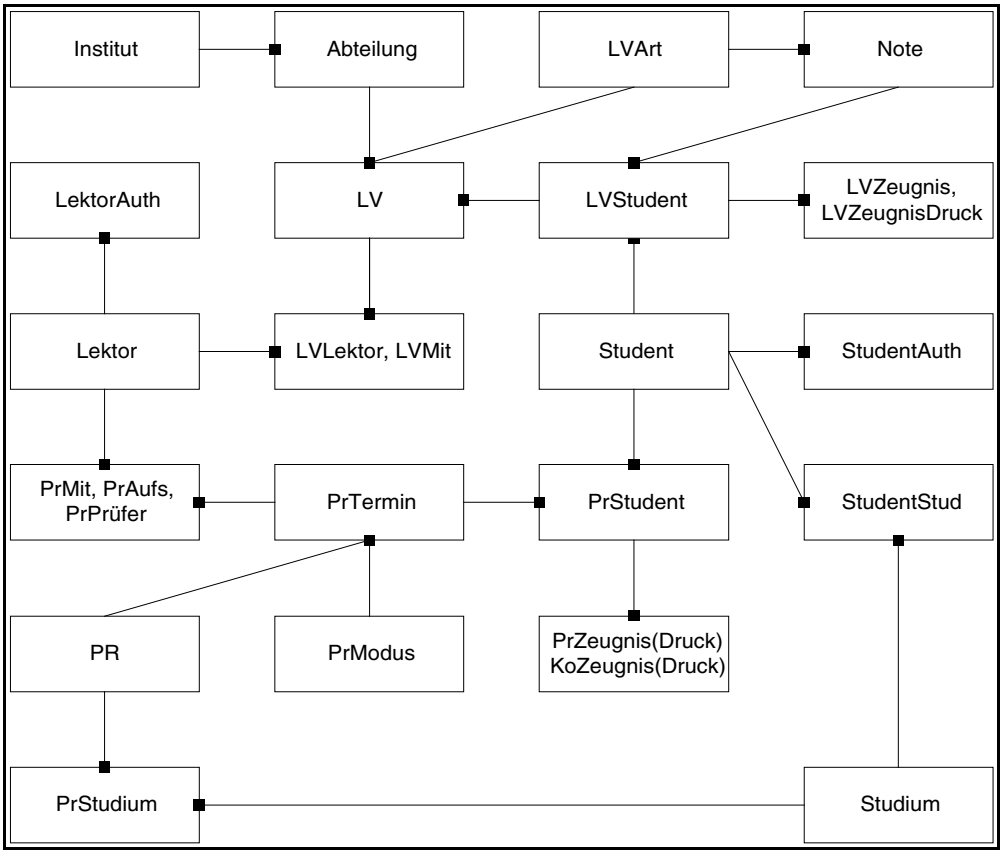

Abb. 8: Datenmodell des alten IIS

Legende:

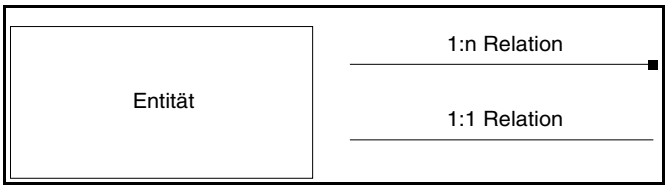

Abb. 9: ER-Diagramm - Legende

In diesem Modell wurden gleichartige Tabellen zusammengefaßt. So wurden beispielsweise die Tabellen „PrZeugnis“ und „PrZeugnisDruck“ (die Informationen über Prüfungszeugnisse vor und nach dem Ausdruck enthalten) in derselben Entität zusammengefaßt. Außerdem wurden, um das gesamte logische Datenmodell zu zeigen, n:m-Relationen aufgelöst (Beispiel: die Relation zwischen LV und Lektor).

Nach dem Hinzufügen der für das Whiteboard und die Publikationenverwaltung notwendigen Entitäten, stellt sich das ER-Diagramm folgendermaßen dar (neue Entitäten sind grau unterlegt):

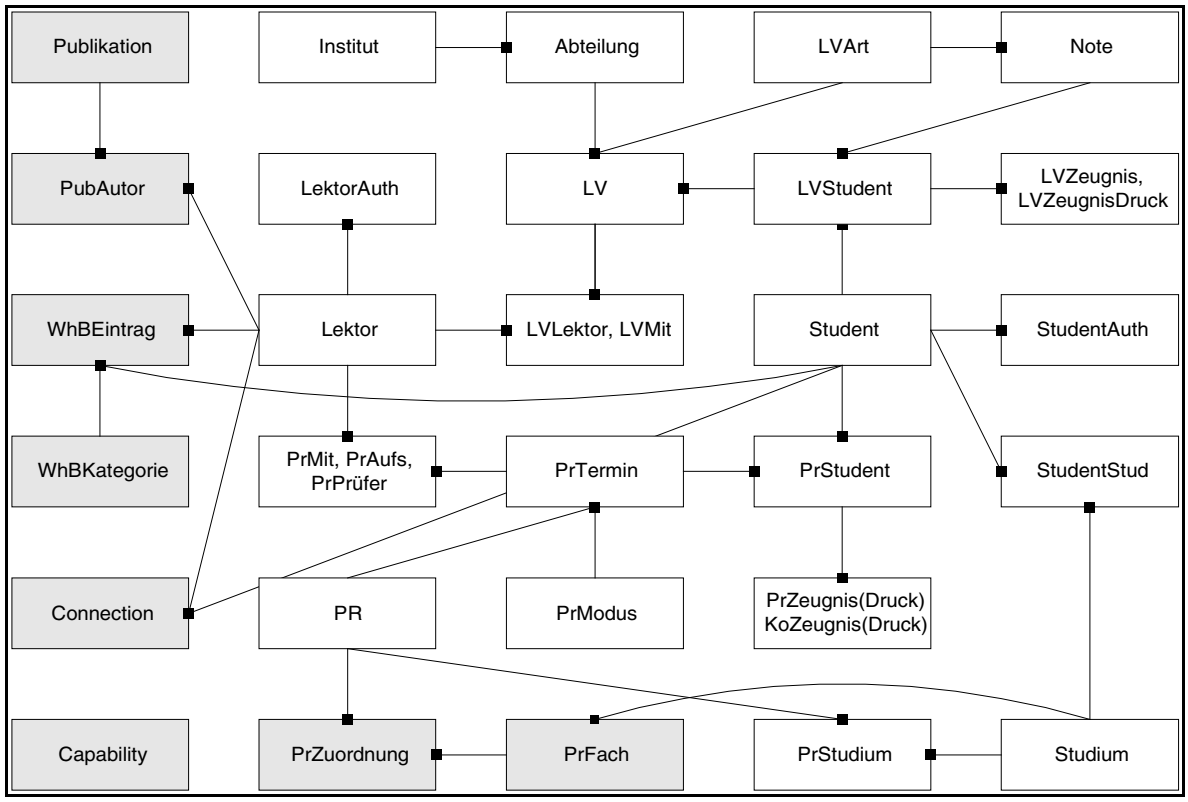

Abb. 10: Datenmodell mit Whiteboard und Publikationen

In der Tabelle „Publikation“ werden die Stammdaten der Publikationen (Titel, Datum, ...) abgelegt, „PubAutor“ stellt den Link zwischen Publikationen und Lektoren dar. Da Publikationen auch meist institutsfremde Autoren enthalten (die im IIS nicht eingetragen sind), enthält dieses Schema auch ein Attribut für einen Namenseintrag.

„WhBEintrag“ enthält die Whiteboardeinträge, incl. einen Verweis auf eine Whiteboard-Kategorie, abgelegt in „WhBKategorie“. In „PrFach“ sind die Diplomprüfungsfächer der Studien abgelegt und in „PrZuordnung“ Prüfungen zugeordnet.

„Connection“ und „Capability“ schließlich werden zur Userverwaltung und zum Datenschutz verwendet.

Das alte IIS verwendet als DBMS Oracle 6. Die Neuentwicklung verlangte aber, nach der Entscheidung für Perl und DBI (s. u.) die Verwendung von Oracle 7. Außerdem bietet Oracle 7 für zukünftige Entwicklungen bessere Perspektiven. Also wurde für das Webinterface ein Database-Link von der Oracle 6 zur Oracle 7-Datenbank hergestellt. Übernommene Tabellen werden über den Link (durch SQL-Net realisiert) angesprochen, die neuen Tabellen werden unter Oracle 7 verwaltet.

2.4 Modellbildung

Die Modellierungsphase beginnt mit der Erstellung eines Benutzerobjekte-Modells. Anschließend wird versucht, möglichst viele Eigenschaften herauszufiltern, die mehreren Tasks gemeinsam sind und sie in Algorithmen darzustellen. Darauf folgt die Systementscheidung, die unter Umständen Rückwirkungen auf die

vorhergegangenen Phasen hat. Am Ende der Modellierungsphase steht die Formulierung eines Systemobjekte-Modells. Graphisch stellt sich diese Phase folgendermaßen dar:

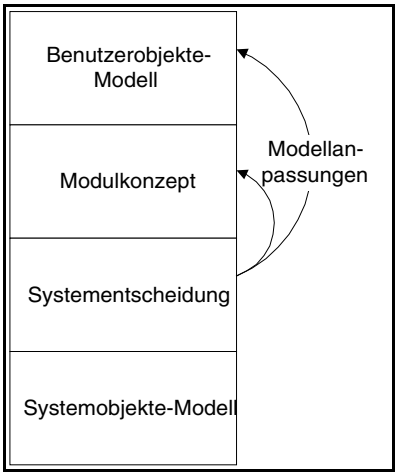

Abb. 11: Modellierung

Das Benutzerobjekte-Modell fürs IIS hat als Objekte hauptsächlich die verschiedenen Benutzergruppen und ihre primären Tasks. In ER-ähnlicher Notation (das ist nicht die von Redmond/Moore vorgeschlagene, aber eine leicht verständliche!) dargestellt, kann es z. B. so aussehen:

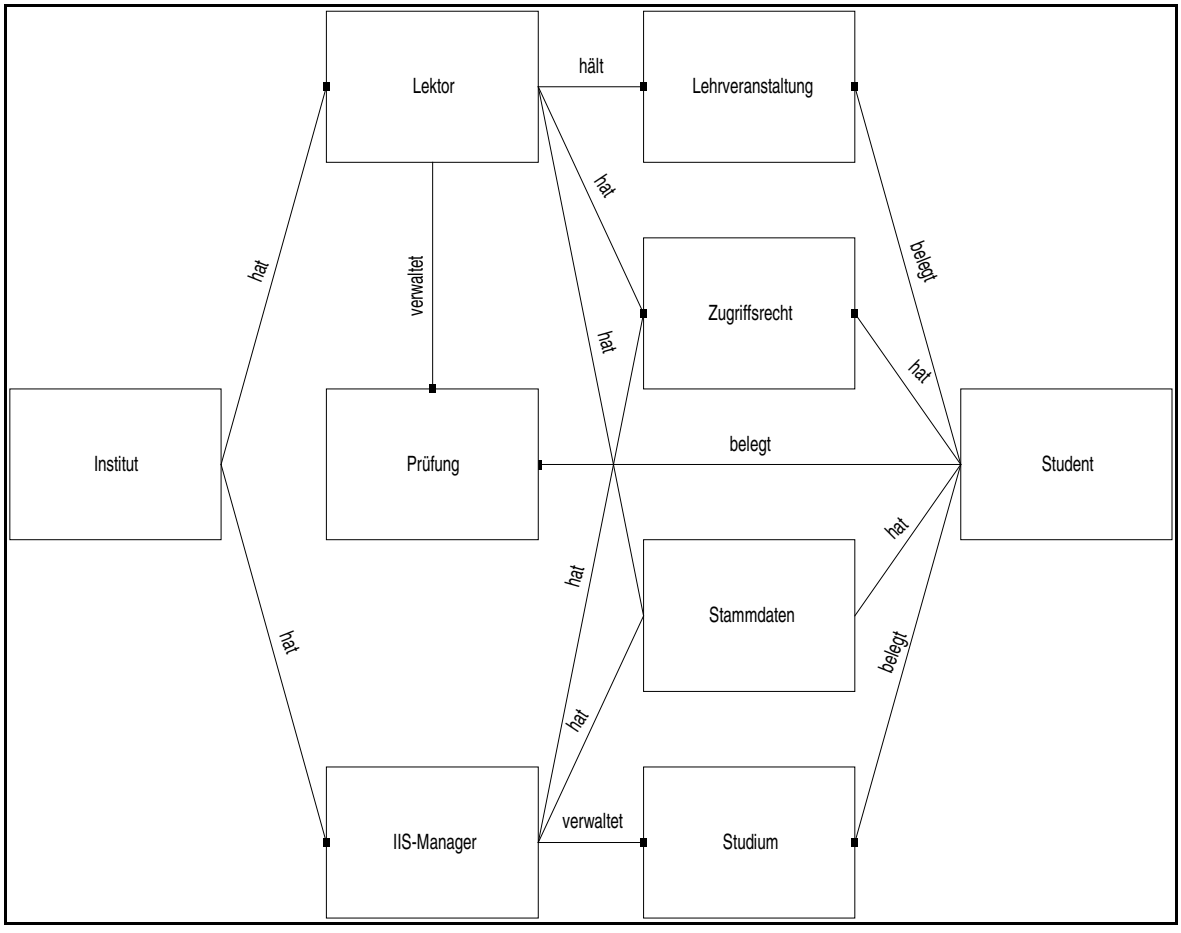

Abb. 12: Benutzerobjekte-Modell

Dabei stellen Rechtecke Objekte und Linien, wie bei ER-Diagrammen, Beziehungen dar. So bedeutet beispielsweise die Beziehung zwischen Lektor und Prüfung, daß ein Lektor n Prüfungen verwaltet und nicht umgekehrt, wie man das von einem eher resignativen Standpunkt aus sehen könnte. Der Hauptzweck des Benutzerobjekte-Modells ist, die Problemstellung aus Benutzersicht möglichst klar darzustellen; Details wurden also nicht berücksichtigt.

Großes Gewicht kommt der Modulkonzept-Phase (MKP) zu. Sie ersetzt im wesentlichen die frühere Feinprojektierung nach dem Wasserfallmodell. Um die beiden Hauptziele, die Vermeidung von Doppelgleisigkeiten in der Programmierung und die Vorbereitung von Bausteinen für das Prototyping zu erreichen, ist es nötig, die zuvor festgestellten Tasks und Subtasks zu vereinheitlichen und möglichst viele Gemeinsamkeiten herauszufiltern.

Betrachtet man die Taskliste aus 2.2, so ergeben sich im wesentlichen vier Arten von Tasks:

Nr.	Tasktyp	Beispiel
1	Informationen lesen	Prüfungsinformationen lesen
2	Informationen hinzufügen	Zu Prüfungen anmelden
3	Informationen ändern	Paßwort ändern
4	Informationen löschen	Von Prüfungen abmelden

Tab. 6: Tasktypen

Alle diese Tasks bestehen aus eine Folge von drei Programmfunktionen, die alle eine Entsprechung in der Datenbank haben:

#	Funktion	SQL-Entsprechung
1	Liste ausgeben	SELECT über n Sätze
2	Maske mit einem (oder keinem) Datensatz ausgeben	SELECT über einen Satz
3	Datenbankmanipulation durchführen	INSERT, UPDATE, DELETE

Tab. 7: Programmfunktionen

Analysiert man die Tasktypen nach ihrer Zusammensetzung aus den drei Programmfunktionen, so ergibt sich meistens folgender Aufbau:

Tasktyp	Aufbau
Informationen lesen	1. Ausgabe einer Liste von Datensätzen 2. Ausgabe eines ausgewählten Datensatzes in Detailform
Informationen hinzufügen	1. Ausgabe einer leeren Maske 2. Hinzufügen der eingegebenen Informationen
Informationen ändern	1. Ausgabe einer Liste von Datensätzen 2. Ausgabe einer Maske mit einem ausgewählten Datensatz 3. Durchführen der Benutzeränderungen in der Datenbank

Informationen löschen	1. Liste von Datensätzen ausgeben 2. Löschen des selektierten Datensatzes

Tab. 8: Aufbau der Tasktypen

Um die Benutzbarkeitsanforderung des direkten Feedbacks einhalten zu können, ist es notwendig, noch die Programmfunktion „Ausgabe einer Bestätigung/Fehlermeldung“ hinzuzufügen. Schließlich ist für den Login-Vorgang (zu dessen besonderer Problematik siehe 3.3) aus Sicherheitsgründen noch eine eigene Programmfunktion vorzusehen.

Nachdem sichergestellt ist, daß sich alle Tasks mithilfe dieser nunmehr fünf Programmfunktionen realisieren lassen, sind diese nach Gemeinsamkeiten zu untersuchen. Es ergeben sich die folgenden Anforderungen:

#	Anforderung
1	Die Funktionen zur Listen- und Maskenausgabe verwenden beide Schablonen.
2	Diese Schablonen müssen variable Teile enthalten können, die durch Datenbankwerte bzw. Benutzereingaben ersetzt werden.
3	Die verschiedenen Datenbankfehler müssen vereinheitlicht und mit variablen Fehlermeldungen behandelt werden können.
4	Alle Ausgaben erfolgen - im Aufbau - nach der später im Style-Guide festgelegten Art.
5	Auszuführende SQL-Statements müssen ebenfalls variable Teile enthalten können.

Tab. 9: Anforderungen an Programmfunktionen

Die Programmfunktionen müssen also so realisiert werden, daß sie konfigurierbar sind. Es müssen unterschiedliche Schablonen (oder Vorlagen) bzw. Verweise auf Vorlagen angegeben werden können, usw. Realisiert man die Programmfunktionen als Funktionen eines Client-Programmes, so kann die Steuerung über Parameter erfolgen, entscheidet man sich dafür, die Programmfunktionen als selbständige Applikationen zu realisieren, die von Benutzeraktionen aktiviert werden und ihrerseits wieder Benutzereingaben ermöglichen, so kann die Steuerung entweder über Kommandozeilenparameter oder Konfigurationsscripts erfolgen.

Die Entscheidung für die konkrete Entwicklungsumgebung ist der nächste Schritt der Modellierung. Da beim IIS das Betriebsystem (UNIX) und die Netzwerkumgebung (INTERNET) vorgegeben waren, beschränkte sich die Entscheidung auf das Kommunikationsmedium zwischen dem Benutzer und der Datenbank. Zur Auswahl standen die Alternativen JAVA mit Oracle-Schnittstelle oder Perl mit HTML und Oracle-Schnittstelle. JavaScript schied aufgrund mangelnder Sicherheit von vornherein aus.

Für JAVA sprach die kompakte Präsentation des Client-Systemes, ev. höhere Performance als bei Perl-CGI-Scripts (z. B. durch Vermeiden mehrfacher Datenbanklogins) und die mit dem Namen verbundene Euphorie. Perl dagegen lockte mit kürzeren Entwicklungszeiten, integrierter Fehlertoleranz (ein Motto von Perl ist, daß man jeden Code interpretieren kann ...) und der besseren Eignung für

Prototyping (z. B. daß man das System auch verwenden kann, während daran gearbeitet wird). So fiel die Entscheidung auf die Verwendung von Perl-CGI-Scripts mit HTML und dem Perl-CGI-Package als Benutzerschnittstelle und dem Perl-DBI/DBD-Package als Oracle-Schnittstelle.

Der letzte Teil der Modellierungsphase war die Erstellung eines Systemobjekte-Modells, das die Problemstellung aus Systemsicht möglichst klar darstellen soll:

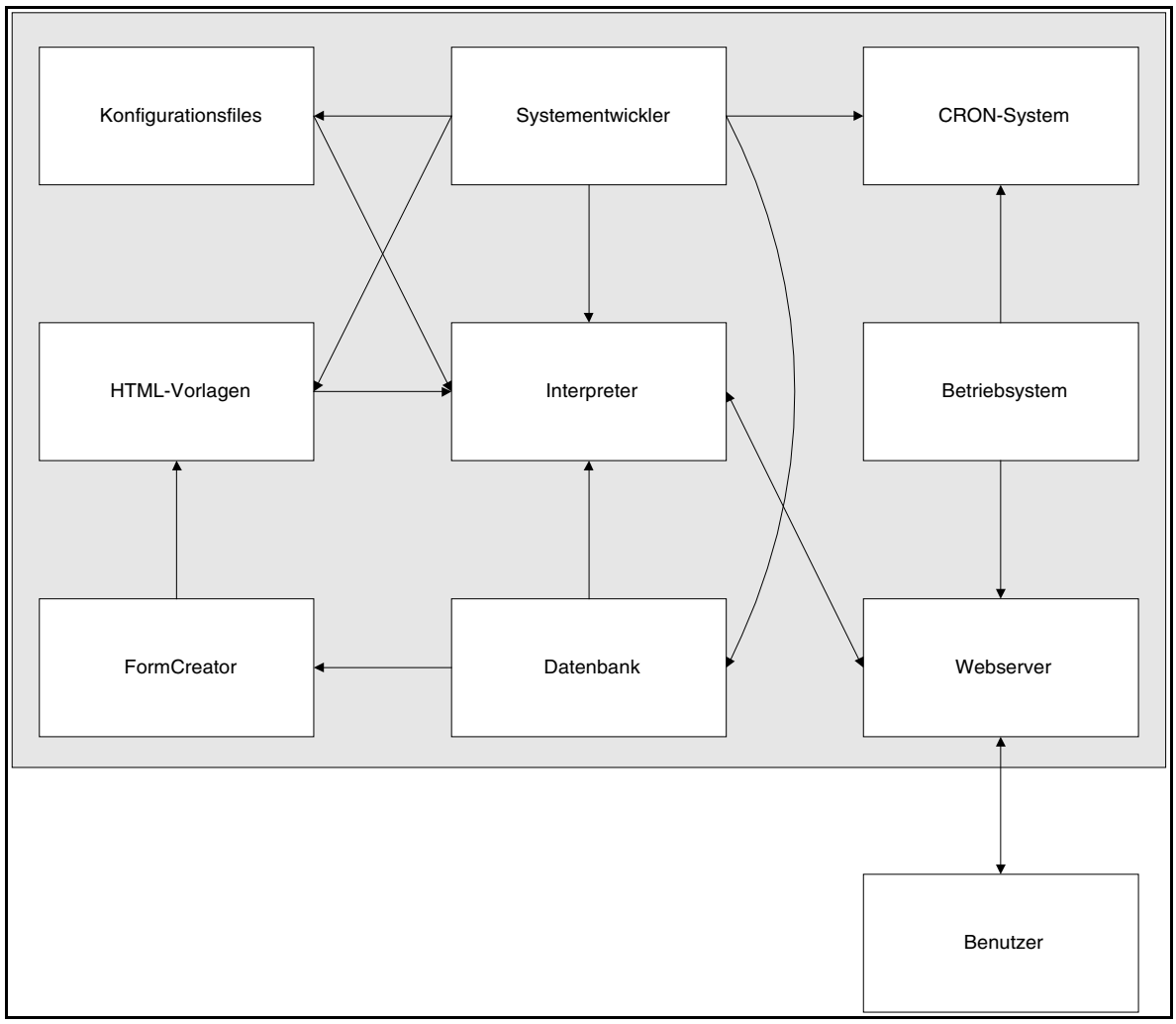

Abb. 13: Systemobjekte-Modell

Rechtecke bezeichnen Systemobjekte und Pfeile Einflüsse.

2.5 Realisierung

Im Rahmen der Realisierung wird zuerst der Style-Guide festgelegt. Anschließend werden die zuvor festgestellten Module entweder als eigenständige Programme oder als Bibliotheksfunktionen programmiert und ausgetestet. Danach beginnt das Prototyping mit dem gruppenweisen Design der zu realisierenden Tasks und der darauffolgenden Freigabe für die Evaluation durch die Testuser:

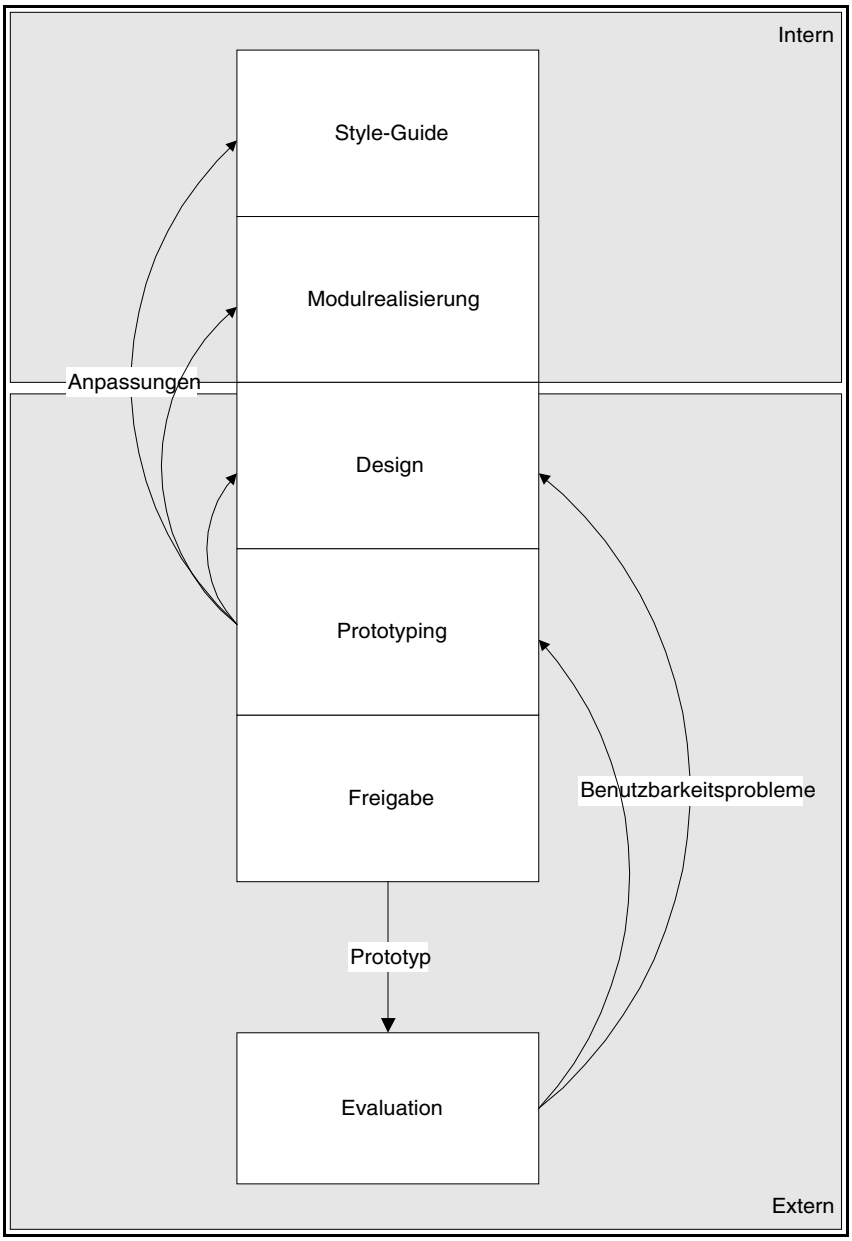

Abb. 14: Kontinuierliches Prototyping

Aus Sicht des Systementwicklers erfolgt die Modulrealisierung intern, während die Taskprogrammierung externen Zwecken dient.

Der IIS-Style-Guide wurde möglichst einfach festgelegt: Grafiken wurden möglichst sparsam benutzt (als Logo, Hintergrund und für die Buttonleiste). Eine Buttonleiste wurde als eigener Frame am unteren Ende des Bildschirmes angefügt, um das schnelle Wechseln zu den Hauptseiten des IIS (entsprechend dem WWW-Konzept) zu ermöglichen. Jede Seite hat einen Seitenkopf mit dem Logo auf der linken und einer Überschrift auf der rechten Seite, abgeschlossen von einem Absatzzeichen. Am Ende jeder Seite befindet sich wiederum ein Absatzzeichen, ein zentrierter Link zur Benutzer-Hotline und (noch) ein Verweis zu Homepage des Herstellers. Eine IIS-Seite sieht beispielsweise so aus:

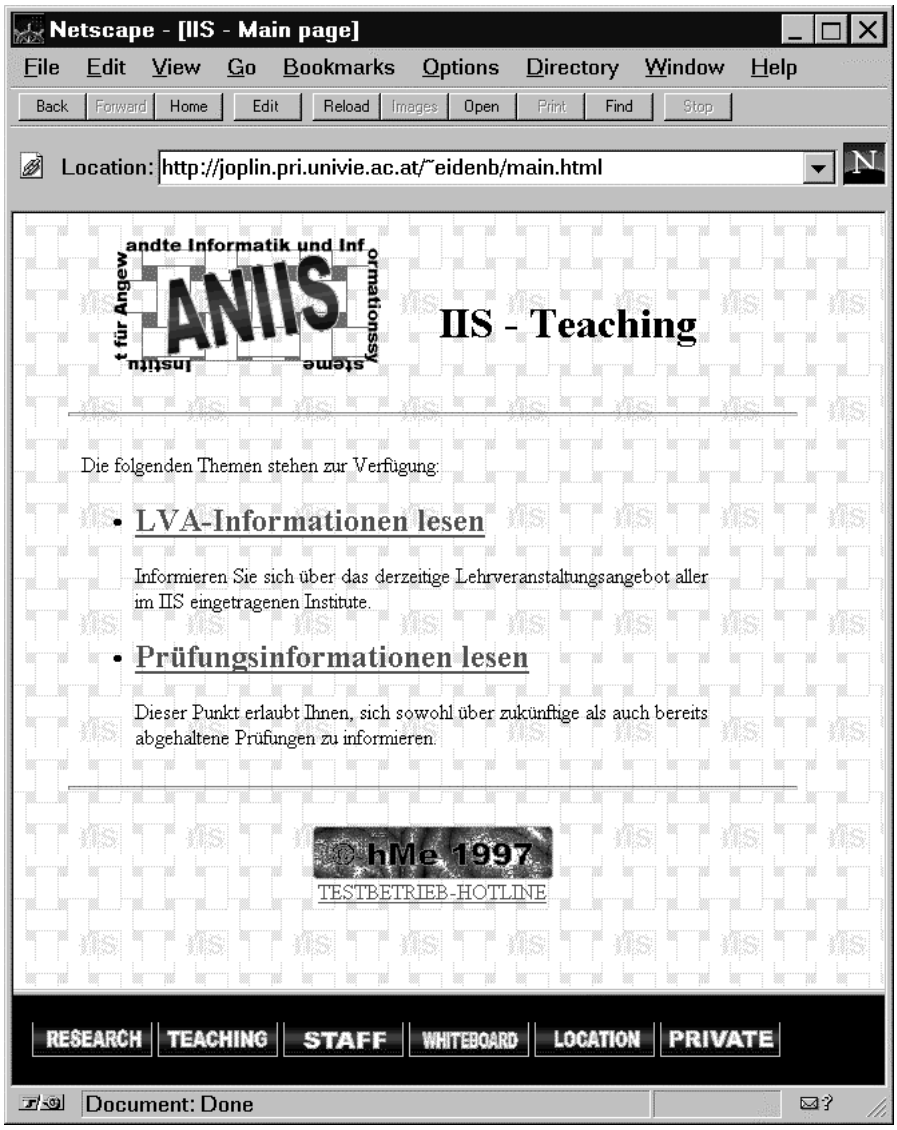

Abb. 15: Style-Guide - Screenshot

Weiters wurde versucht, durch Verwendung von Vorlagen für HTML-Seiten die Seitengestaltung möglichst einheitlich zu halten.

Die Modulrealisierungsphase begann mit der Erstellung einer Reihe von Bibliotheksfunktionen (wie z. B. Seitenkopf ausgeben, Login-Status überprüfen, usw.), die in einem Perl-Package (IIS.pm) zusammengefaßt wurden. Anschließend wurden die Programmfunktionen als selbständige Applikationen realisiert, die - je nach Aufgabe - Bildschirmausgaben oder Datenbankmanipulationen durchführen und durch Konfigurationsfiles mit einer eigens festgelegten Syntax konfiguriert werden, weshalb sie in der Folge Interpreter(-Scripts) genannt werden. Diese Syntax ist im Kapitel 4.3 beschrieben.

Danach begann mit der Realisierung der Login-Funktion für Studenten die erste Prototypingphase. Das WWW-Konzept äußert sich nicht zu der Problematik, daß gewissen Funktionen (wie z. B. Prüfungsanmeldung, - status lesen, usw.) nur (durch Paßwort) autorisierten Personen zur Verfügung stehen dürfen. Gegenüber der Möglichkeit, bei jeder Aktion das Paßwort abzufragen, wurde jene vorgezogen, die

geschützten Funktionen in einer Shell zusammenzufaßt, welche nur durch die korrekte Eingabe des Paßwortes erreicht werden kann.

Das eigentliche Prototyping bestand nach der Modulrealisierung (von Änderungen abgesehen) nurmehr im Formulieren von Konfigurationsfiles für die Interpreter. Diese Konfigurationsfiles enthalten hauptsächlich SQL-Statements für den Datenbankzugriff und HTML-Text für die diversen Bildschirmausgaben (Vorlagen, Titel, Fehlermeldungen, usw.). Vorlagen können auf zwei Arten definiert werden, als String im Konfigurationsfile oder als Verweis auf eine Textdatei, was besonders bei Ein- und Ausgabemasken von Vorteil ist.

Zum Generieren von Eingabemasken aus Datenbanktabellen wurde eine simple Applikation entwickelt (der FormCreator), die sich als sehr hilfreich erwies. Die Links, mit denen CGI-Scripts in HTML-Pages des IIS eingefügt wurden, bestanden aus den Interpreter-Scripts und ihren Parametern (v. a. dem Pfad des Konfigurationsfiles, aber auch der User-ID, usw.).

Diese Art des Prototyping erwies sich als sehr effizient, sodaß die Taskliste in der Hälfte der vorher veranschlagten Zeit abgearbeitet werden konnte. Da die Interpreter (und das dahinterstehende Konzept) nicht IIS-spezifisch sind und die dahinterstehende Datenbank austauschbar ist, läßt sich diese Entwicklungsumgebung, zusammen mit dem Vorgehensmodell des kontinuierlichen Prototypings zur Entwicklung aller denkbaren Webinformationssysteme anwenden, was ein netter Nebeneffekt der IIS-Erweiterung ist.

2.6 Evaluation

Parallel zur IIS-Entwicklung erfolgte nach der Freigabe der entsprechenden Komponenten, die Evaluation durch die Testuser. Aus Sicht der Benutzer sind die freigegebenen Komponenten lediglich als (möglichst einfacher und robuster) Lösungsvorschlag zu sehen, für dessen detaillierte Ausformulierung sie selbst verantwortlich sind.

Indem sie ihre Änderungswünsche dem Systemplaner mitteilen, können sie auf sehr hohem Niveau die Gestaltung ihres Systemes vornehmen. Das hat für den Systementwickler den Vorteil, daß er im Nachhinein negatives Feedback mit dem Verweis auf die Selbstverantwortlichkeit der Benutzer abweisen kann. Diese Argumentation ist jedoch nur in dem Maße zulässig, wie der Lösungsvorschlag der als optimal angesehenen Lösung entspricht.

Die Evaluation der IIS-Funktionen wurde von den Dozenten und Assistenten des Institutes für Angewandte Informatik und Informationssysteme selbst übernommen. Änderungswünsche wurden von verschiedenen Seiten geäußert, beschränkten sich aber zumeist auf Kleinigkeiten (wie Rechtschreibfehler, Layoutänderungen, usw.).

Komplexere Änderungen, wie z. B. die Kategorisierung von Prüfungen nach dem Prüfungsfach, konnten oft aufgrund der Einschränkung, das bisherige Datenmodell weitgehend beizubehalten (um alte SQL-Forms-Anwendungen weiter benutzen zu können), nicht erfüllt werden. Im konkreten Fall der Prüfungen konnte dieser Nachteil

aber durch eine Suchfunktion annähernd ausgeglichen werden. Folgende Änderungen wurden durchgeführt:

Nr.	Benutzer	Änderungen
1	Doz. Dr. Breiteneder	Layout, LVA-Anmeldung
2	Doz. Dr. Hitz	Tippfehler, Suchfunktionen
3	Dr. Polaschek	Logo
4	Doz. Dr. Breiteneder	Zusätze
5	ANIIS-Mitarbeiter	LVA-Anmeldung, Verschiedenes

Tab. 10: Änderungen

Als Hauptmangel des Systemes erwies sich - wie erwartet - die eher schlechte Performance von Scripts, die ihre Daten direkt aus der Datenbank nehmen. Diesem Mangel konnte durch die automatische, periodische Generierung von Seiten (mithilfe des CRON-Systemes) begegnet werden. Zur Generierung werden die Interpreter-Scripts verwendet, deren Ausgabe vom WWW-Demon auf ein File umgeleitet wird. Außerdem mußten Hypertext-Links auf generierte Seiten umgeändert werden.

2.7 Installation

Die Entwicklung des IIS-Webinterface erfolgte auf einem Studentencluster des Institutes unter Verwendung einer Oracle 7-Datenbank mit einem Link zur Testdatenbank des Institutes unter Oracle 6.

Zur Anbindung des Systemes an die Produktionsdatenbank wurde ein Installationsplan erstellt, der im wesentlichen folgende Aktivitäten vorsieht:

Nr.	Aktivität
1	Voraussetzungen schaffen (Oracle 7, ...)
2	Oracle 7 - Datenbankumgebung erstellen
3	Database Link zur Produktionsdatenbank herstellen
4	Unix-Umgebung erstellen
5	Entwicklungsumgebung, Interpreter und Vorlagen installieren
6	Neue IIS - Tabellen anlegen
7	Seitengenerierung im CRON-System eintragen
8	Systemtest

Tab. 11: IIS - Installation

Dieser Ablauf ist größtenteils nicht IIS-spezifisch, ergibt sich also bei den meisten ähnlichen Aufgabenstellungen.

2.8 Zusammenfassung / Projekttagebuch

Das zweite Kapitel beschrieb detailliert die Abarbeitung des Vorgehensmodelles. Zuerst wurde die Frage beantwortet, welche Benutzer das System benutzen werden. Anschließend wurden die Nebenbedingungen festgelegt, unter denen diese Benutzung geschieht, bevor die Aufgaben bis zu einem sinnvollen Niveau detailliert

wurden. Danach begannen (parallel) die Datenmodellierungsphase und die (Programm-)Modellierung. Ein Datenbankschema wurde gebildet, Systemmodelle aus Benutzer- und Systemsicht entworfen, Gemeinsamkeiten zwischen Aufgaben gesucht und modelliert und schließlich die Entwicklungsumgebung festgelegt.

Danach begann mit der Style-Guide-Phase und der Realisierung der Prototypingbausteine die Realisierung des Systemes. Das eigentliche iterative Prototyping arbeitete dann den Task- (oder: Prototyping-) Plan ab; Prototypen wurden aus den vorher erstellten Bausteinen zusammengebaut und für die Evaluation freigegeben. Während der Evaluationsphase hatten die Testbenutzer die Möglichkeit, die Applikation nach ihren Wünschen zu gestalten, bevor das System installiert und abgenommen wurde.

Für die IIS-Aufgabenstellung wurden folgende Aktivitäten (in den angegebenen Zeiten) durchgeführt:

1. Analysephase (8.7. - 8.8.)

Datum	Meilenstein
8.7.	Projektbeginn
10.7.	Besprechung der Aufgabenstellung
14.7.	Zeitplan steht
16.7.	Security/CASE-Überlegungen
23.7.	Datenbankschema erhalten
29.7.	Entwicklungsumgebung eingerichtet
8.8.	Analysephase abgeschlossen, Modulkonzept steht

Tab. 12: Analysephase

2. Realisierungsphase (9.8. - 11.9.)

Datum	Meilenstein
15.8.	Modulrealisierung abgeschlossen
19.8.	Testdatenbank eingerichtet
	Erweiterter Prototyping-Plan erstellt
20.8.	Datenbank-Besprechung
	Lösung des Oracle-Versionsproblemes
	Neuer PT-Plan erstellt
28.8.	Erste PT-Phase abgeschlossen
1.9.	CB Änderungen
2.9.	PT Phase 2,3 abgeschlossen
3.9.	Hitz-Änderungen
8.9.	Lösung des Staff-Problemes
11.9.	Abschlußbesprechung Realisierung

Tab. 13: Realisierungsphase

3. Programmbeschreibung

3.1 Konzepte

Im folgenden werden einige Konzepte, die beim Systementwurf berücksichtigt wurden, erläutert.

Die Textsprache wurde entsprechend den Empfehlungen des WWW-Konzeptes gewählt: Englisch für alle öffentlich zugänglichen Seiten außer den für Studenten bestimmten (LVA-Informationen, Prüfungsinformationen, usw.) und Deutsch für die Login-Shells (oder: private Menüs).

Jede Benutzeraktion wird entweder durch die Ausgabe einer Maske, einer Bestätigung oder einer Fehlermeldung beantwortet. Bei den privaten Funktionen (z . B. LVA-Anmeldung) wird die Bestätigung/Fehlermeldung, wo möglich, mit der Menüausgabe kombiniert, um die Abläufe zu verkürzen.

Bei besonders gekennzeichneten Feldern in HTML-Eingabemasken nehmen die Interpreter eine automatische Syntaxüberprüfung vor. Derzeit werden die drei Typen NUMBER (kardinale Zahlen), DATE (Oracle-Datum) und TEXT (unformatiert) unterschieden. Zur Variablentypisierung siehe Kapitel 4.4. Entspricht ein Eingabewert nicht dem angegebenen Typ, so wird eine Fehlermeldung ausgegeben:

Abb. 16: Datentypprüfung - Screenshot

Wie bereits erwähnt, wird an geeigneten Stellen Seitengenerierung verwendet, um den zeitaufwendigen Online-Zugriff auf die Datenbank zu vermeiden. Dazu werden die Interpreter verwendet, deren Ausgabe auf eine Datei umgeleitet wird. Ausgeführt wird die Seitengenerierung periodisch durch das, auf allen Unix-Systemen vorhandene, CRON-System (dzt. täglich um Mitternacht). Der Ablauf der Generierung läßt sich folgendermaßen darstellen:

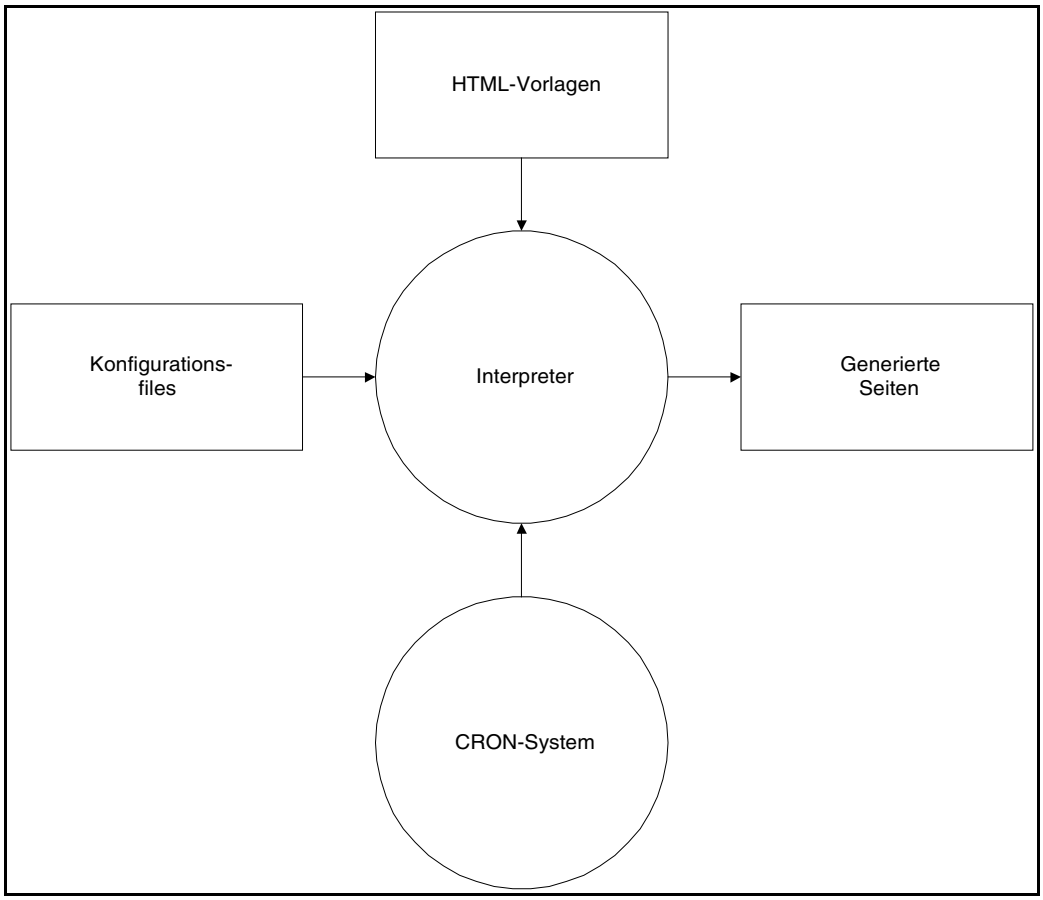

Abb. 17: Seitengenerierung

Schließlich werden dort, wo die IIS-Datenbank nicht ausreichend detailliert ist, Suchfunktionen zum Filtern der Daten verwendet:

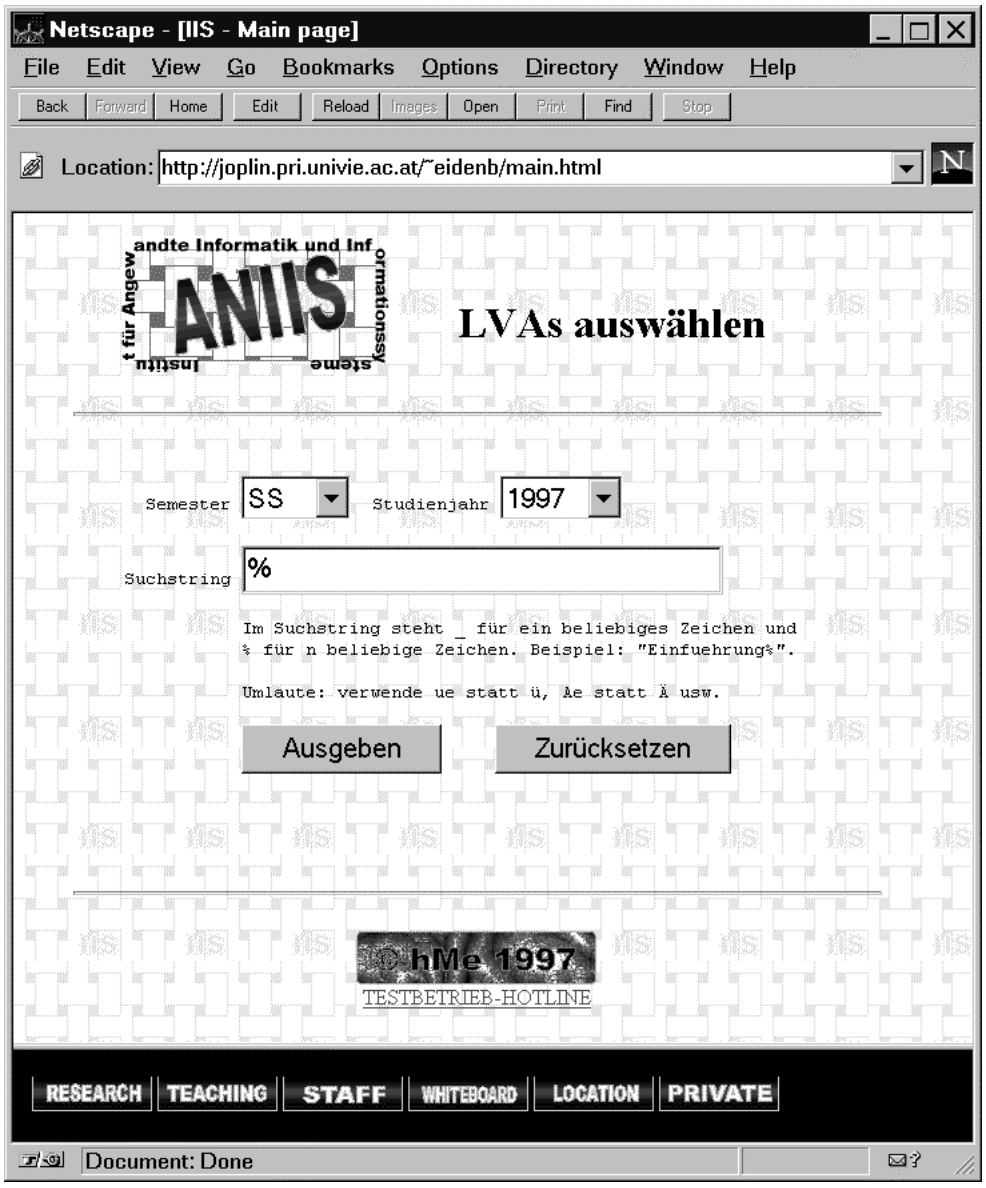

Abb. 18: Suchfunktionen - Screenshot

Die Suchfunktionen stützen sich auf den SQL „like“-Operator, wodurch in Suchausdrücken die folgenden Zeichen als Sonderzeichen behandelt werden:

Zeichen	Bedeutung
_	ein beliebiges Zeichen
%	n beliebige Zeichen

Tab. 14: Sonderzeichen

3.2 Abläufe

Alle IIS-Tasks (und: eigentlich alle Web-Tasks) lassen sich auf vier Abläufe zurückführen: Daten lesen, hinzufügen, ändern und löschen (die letzten drei kann man unter Datenverwaltung zusammenfassen). Dieses Kapitel stellt diese Abläufe sowie ihre Verknüpfung mit den Benutzereingaben und der IIS-Datenbank in Form von Flußdiagrammen dar.

Informationen lesen ist ein zweistufiger Vorgang, der vom Benutzer (wie alle anderen) durch das Verfolgen eines Links initiiert wird:

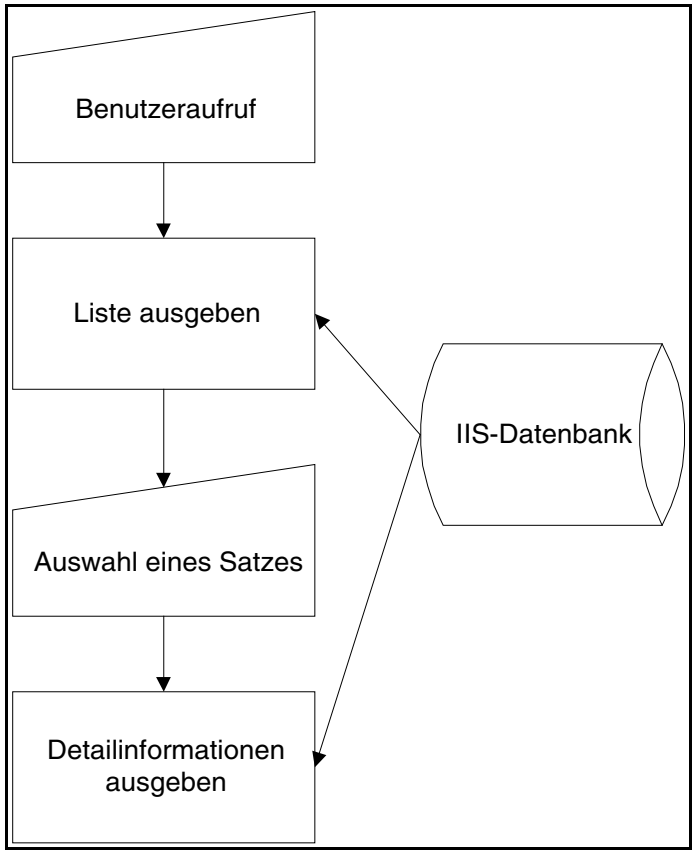

Abb. 19: Informationen lesen - Ablauf

Beispiele für diesen Vorgang sind: Staff-Informationen ausgeben und Publikationen ausgeben. Manchmal, wie bei Lehrveranstaltungsinformationen ausgeben, ist der Listenausgabe auch noch eine Filtermaske (mit Suchfunktion) vorgeschaltet.

Das Hinzufügen von Daten erfolgt ebenfalls nach einem zweistufigen Schema:

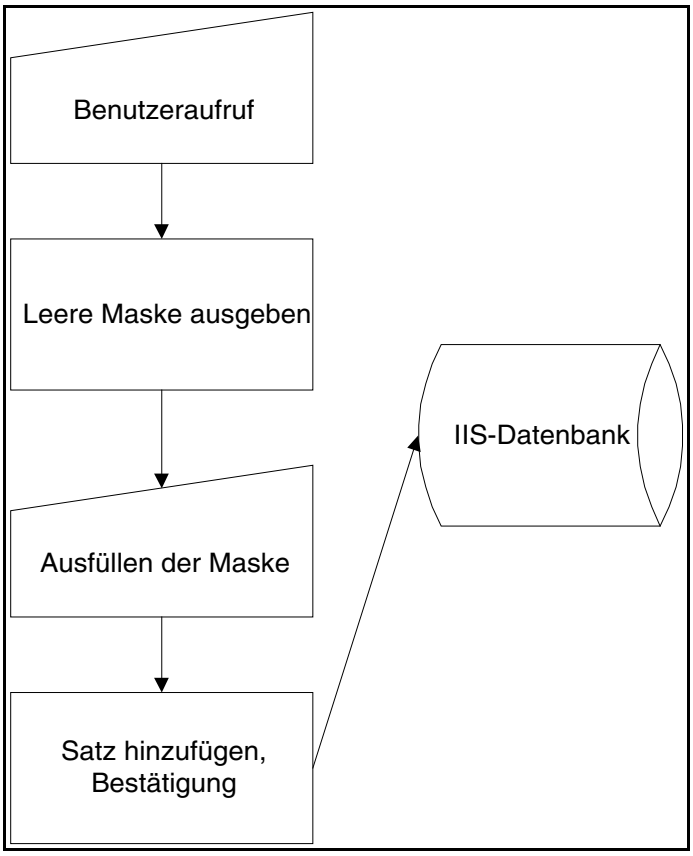

Abb. 20: Informationen hinzufügen - Ablauf

Beispiele für Daten hinzufügen sind: Publikationen hinzufügen, Whiteboardeinträge hinzufügen.

Das Ändern von Daten ist der längste Vorgang, er kombiniert das Eingeben eines neuen Satzes mit dem Löschen eines bereits vorhandenen:

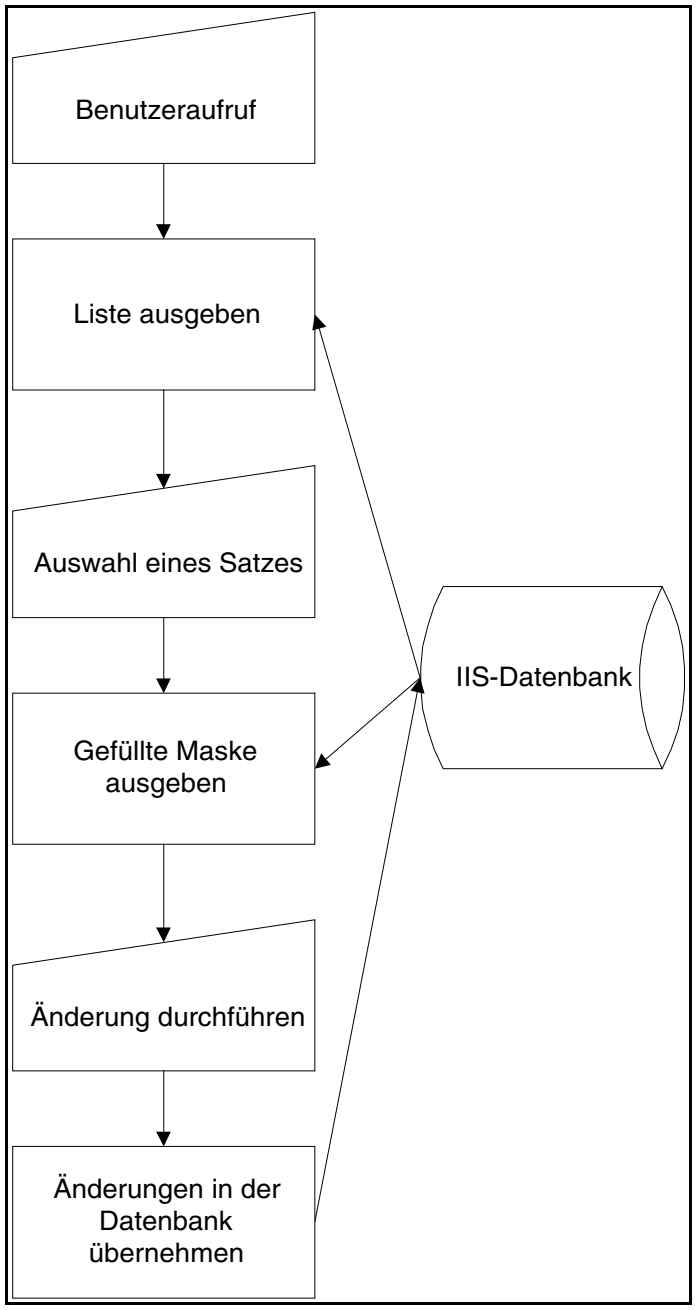

Abb. 21: Informationen ändern - Ablauf

Beispiele für Änderungsfunktionen sind: Whiteboardeinträge ändern, Stammdaten ändern, usw.

Das Löschen von Daten schließlich ist wieder ein zweistufiger Vorgang:

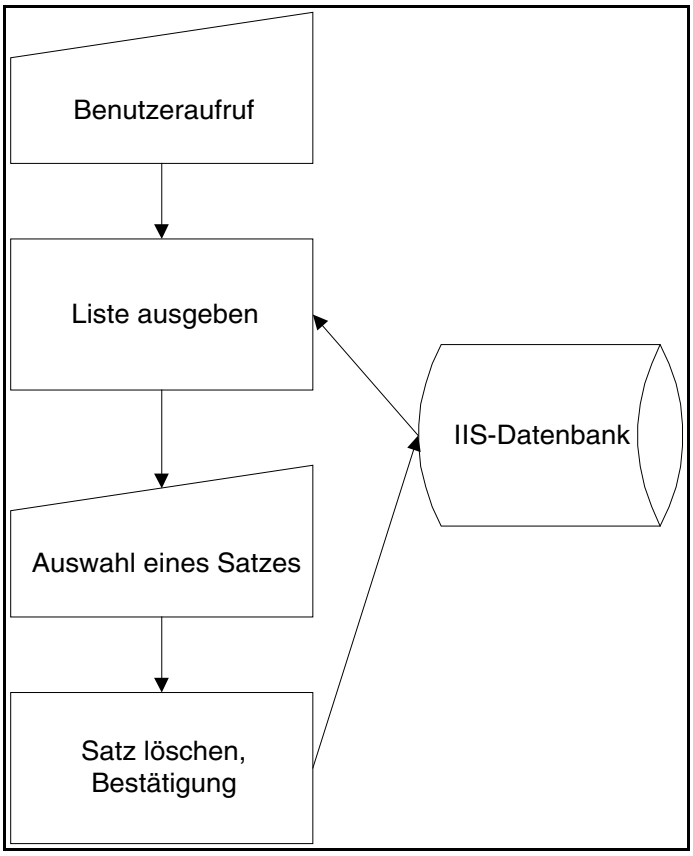

Abb. 22: Informationen löschen - Ablauf

Beispiele sind Whiteboardeinträge löschen oder Publikationen löschen.

3.3 Datensicherheit

Da angenommen werden darf, daß das Institut die Datensicherung von Datenbankinformationen und Unix-Userdaten unter Kontrolle hat, beschränkt sich die Darstellung auf Datenschutzaspekte.

Sowohl auf SQL- als auch auf IIS-Ebene liegt ein diskretes Sicherheitsmodell vor. Es können die Zugriffsrechte (Aktionen) für jedes Subjekt (Benutzer) und jedes Objekt (Tabellenschema) festgelegt werden. Der Nachteil dieses Systems ist, daß es aufgrund seiner Unbequemlichkeit von den Benutzern torpediert wird, indem sie Paßwörter, die den Zugriff regeln, austauschen. Vermutlich wird die DAC (Discretionary Access Control) deshalb als ein Sicherheitssystem niedrigster Kategorie angesehen.

Im neuen IIS gibt es, nach dem Sicherheitslevel, drei Arten von Funktionen: öffentlich zugängliche (Infos lesen, usw.), mit Paßwort zugängliche (LVA anmelden, usw.) und solche, die mit einem Autorisierungsdatensatz des IIS-Systems zugänglich sind (Whiteboard-Zensur für IIS-Manager, ...).

Ein besonderes Problem stellt die Abfrage des Paßwortes dar. Da Webbrowser normalerweise benutzte Seiten cachen, könnte sich, an einem öffentlichen Terminal, ein nachfolgender Benutzer durch das erneute Versenden der Login-Form seines Vorgängers unter einem falschen Namen im System anmelden. Dies läßt sich

vermeiden durch die zweimalige Abfrage und Prüfung des Paßwortes, wobei die zweite Abfrage aus einem Script (und damit bei jedem Aufruf neu) generiert wird. Allerdings verlangt das die zweimalige Angabe das Paßwortes, was unbequem ist.

Die Lösung des neuen IIS für dieses Problem sieht folgendermaßen aus: die Anmeldemaske wird durch ein Script generiert und enthält eine große Zufallszahl, die auch in der Datenbank abgelegt wird. Die vom User übertragenen Login-Daten werden dann auf die Richtigkeit von Paßwort und Zufallszahl hin geprüft. Anschließend wird die Zufallszahl in jedem Fall gelöscht. Stimmt die Zufallszahl und/oder das Paßwort nicht überein, so wird eine Fehlermeldung ausgegeben, andernfalls der Login akzeptiert. Das hat zur Folge, daß man sich mit einer Maske nicht mehrfach einloggen kann. Der Loginablauf sieht folgendermaßen aus:

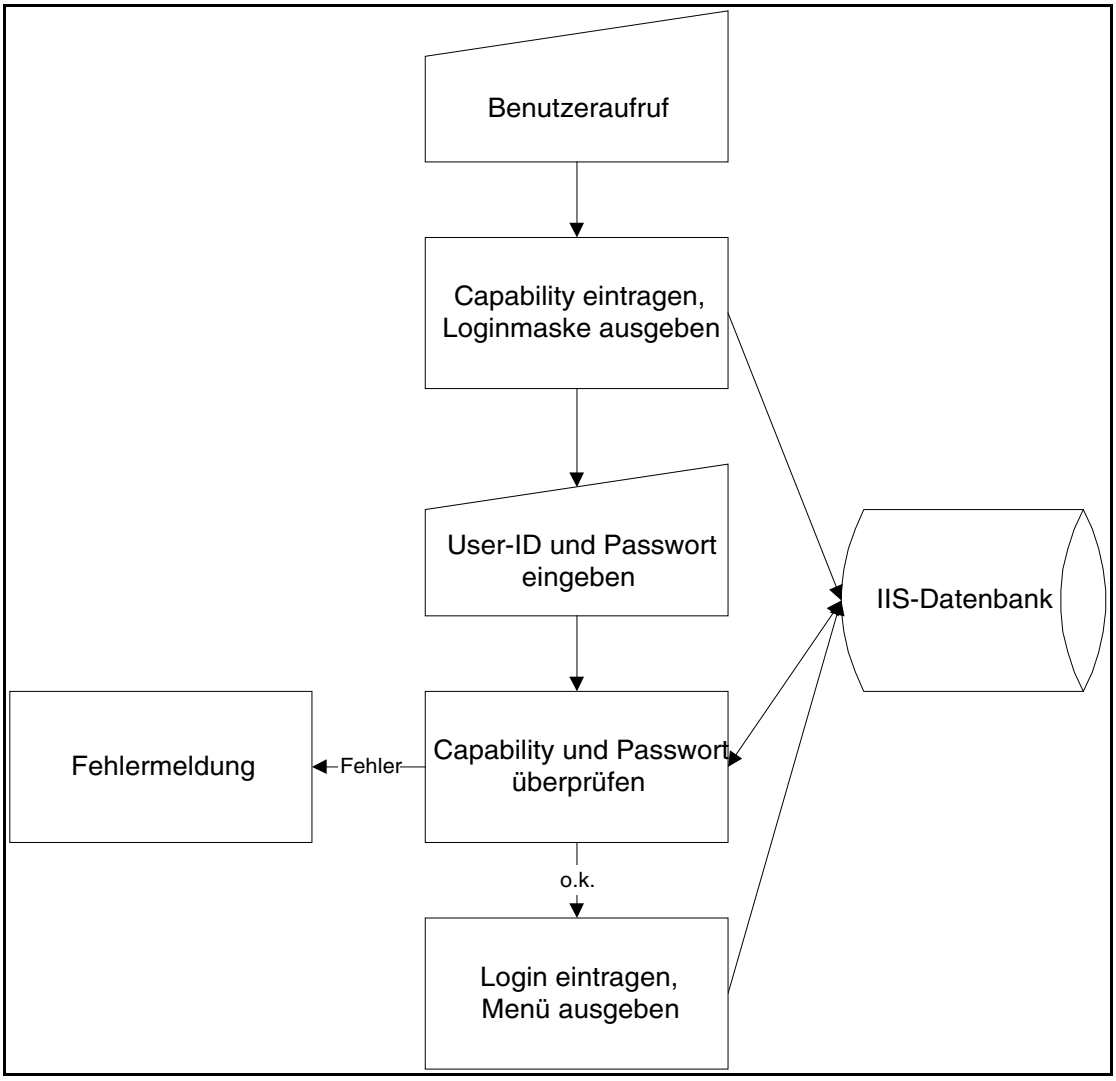

Abb. 23: Login - Ablauf

Ein weiteres Problem wäre, daß nachdem sich ein User ins IIS eingeloggt hat, andere User, die davon Kenntnis haben, auf anderen Geräten Funktionen unter seinem Namen ausführen könnten. Das wird durch die Speicherung und Prüfung der IP-Adresse des Login-Terminals beim Ausführen von sensitiven Funktionen vermieden.

Nach dem Login-Vorgang kommt der Benutzer in eine Shell mit Funktionen, die nur er ausführen darf. Um den Datenschutz zu garantieren, muß er sich aber nach Erledigung aller seiner Arbeiten vom System abmelden (Logout).

Das private Menü enthält alle Funktionen, die zumindest ein Paßwort erfordern. Bei sensitiveren Funktionen (wie der Whiteboard-Zensur) ist außerdem noch ein entsprechender Datensatz in der Tabelle „LektorAuth“ erforderlich (der Rechteverwaltung des IIS).

4. Beschreibung der Entwicklungsumgebung

4.1 Überblick

Die IIS-Entwicklungsumgebung entstand aus dem Wunsch, den Programmieraufwand möglichst zu reduzieren, um so die Zeitrestriktion einhalten zu können. Alle Programme basieren auf Perl. Als Benutzerschnittstelle wurde das CGI-Package verwendet, als Datenbankschnittstelle diente das DBI-Package.

Wesentliches Element der Entwicklungsumgebung sind die Interpreter, die durch Konfigurationsfiles gesteuert werden und ihrerseits auf Vorlagen verweisen, deren variable Teile durch Benutzereingaben oder Datenbankinhalte ersetzt und anschließend ausgegeben werden.

Das Zusammenspiel der einzelnen Komponenten läßt sich folgendermaßen darstellen:

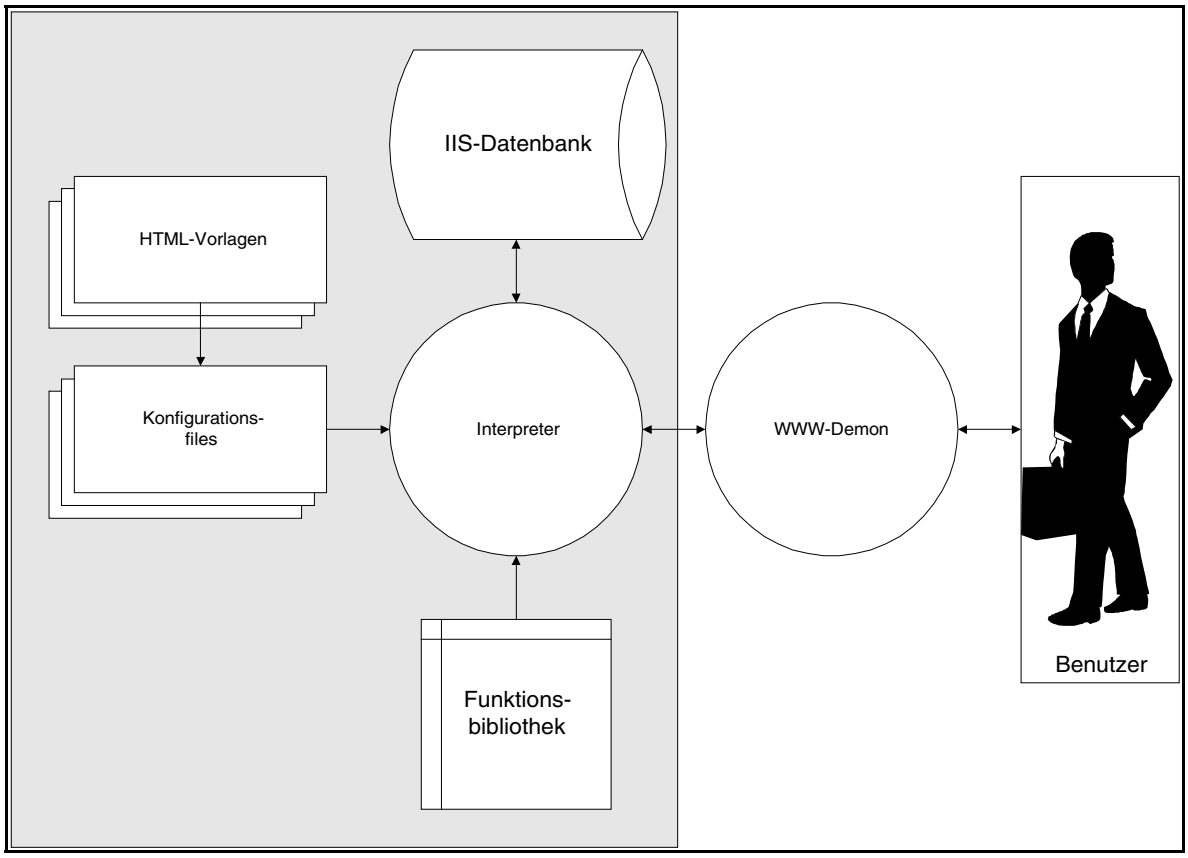

Abb. 24: Zusammenspiel der Systembausteine

Die mehreren Interpretern gemeinsamen Funktionen wurden in einem eigenen Perl-Package abgelegt („IIS.pm“). Im übrigen sind alle Interpreter von einem Standardscript abgeleitet.

4.2 Interpreter-Scripts

Nachdem im Rahmen der Modulrealisierung die Bibliotheksfunktionen erstellt waren, wurde ein Standardscript erstellt, das als Vorlage für die Interpreter diente („standard.vpl“). Dieses Script hat den folgenden Aufbau:

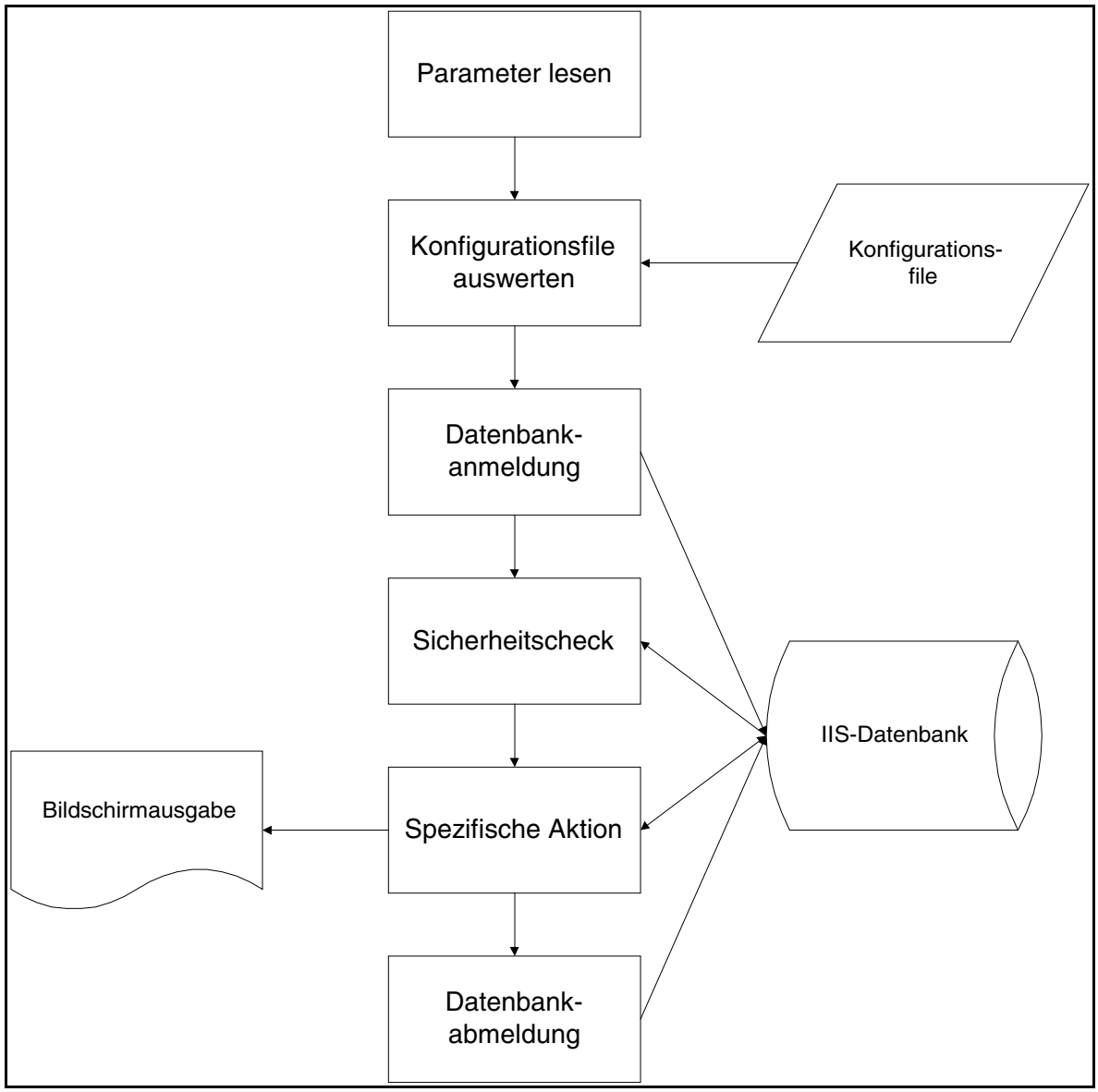

Abb. 25: Aufbau des Standard-Interpreters

Zuerst werden (von der Kommandozeile, der Umgebungsvariable QUERY_STRING, usw.) die Parameter gelesen, anschließend wird die Konfigurationsdatei (auf die die Variable „file“ verweist) ausgewertet. Danach erfolgt die Datenbankanmeldung und in Abhängigkeit vom Sicherheitslevel des Konfigurationsfiles ein Securitycheck. Danach folgt in jedem Script eine spezifische Aktion (Cursor-Select in der Listenausgabe, ...) und anschließend die Datenbankabmeldung.

Die insgesamt sechs verschiedenen Interpreter-Funktionen leiten sich folgendermaßen von der Vorlage ab:

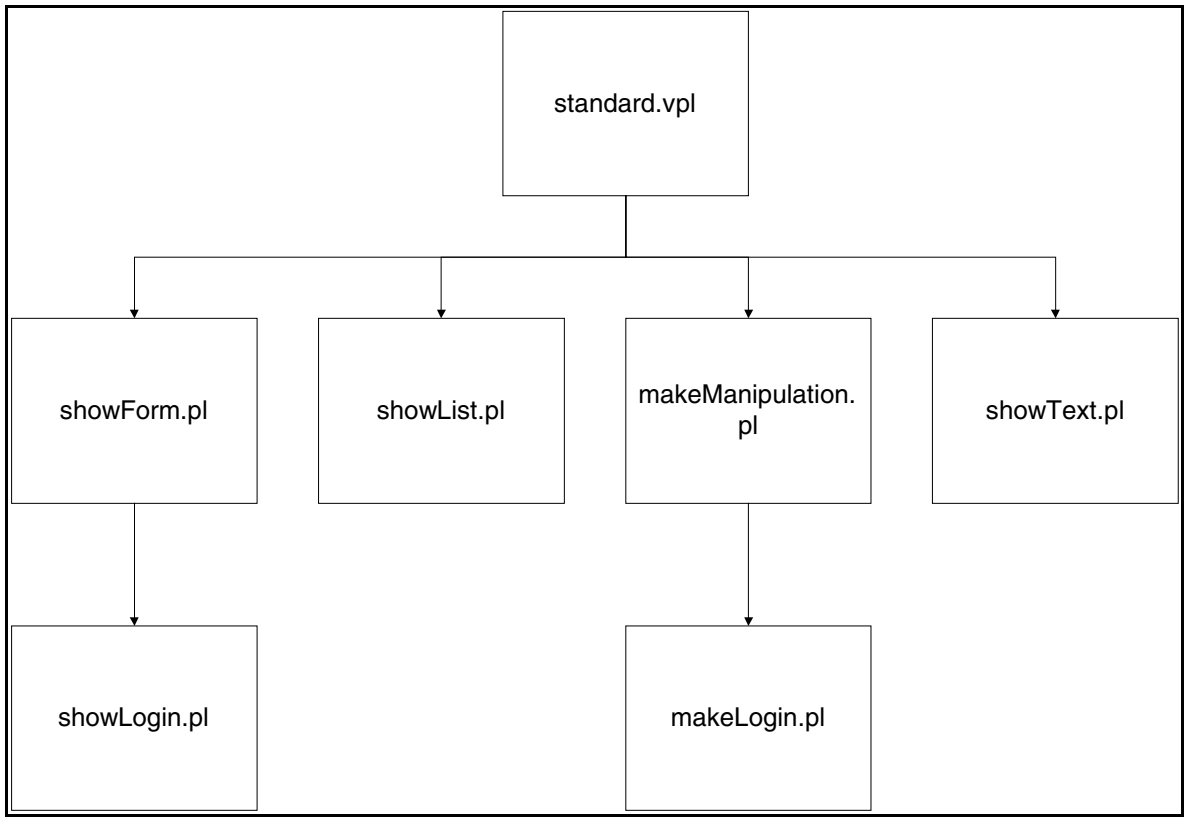

Abb. 26: Interpreter - Ableitungsbaum

„showForm“ („sF.pl“) führt (falls vorhanden) ein Einzelsatz-Select aus, ersetzt anschließend alle Variablen in der Vorlagendatei, führt ev. vorhandene Inline-Statements (s. u.) aus und gibt anschließend die Vorlage aus. „showLogin“ verhält sich ebenso, nur legt es vorher noch einen Eintrag in der Tabelle „Capability“ an und trägt die Zufallszahl in das auszugebende Formular ein.

„showList“ („sL.pl“) führt als spezifische Aktion ein Cursor-Select-Statement und gibt die Ergebniswerte in einer Liste entsprechend einem Vorlagenstring aus.

„makeManipulation“ („mM.pl“) führt Datenbankmanipulationen durch. Es können immer zwei Aktionen in einem Script ausgeführt werden: eine einmalige Datenmanipulation und eine Manipulation, die für ein Array von Parameterwerten (s. u.) wiederholt wird. „makeLogin“ prüft zusätzlich noch die Capability und das Paßwort, legt ev. einen Connection-Datensatz an und gibt eine Meldung aus.

„showText“ („sT.pl“) schließlich gibt eine Textdatei mit dem standardisierten IIS-Kopf- und Fußtext aus.

4.3 Konfigurationsfiles

Konfigurationsfiles steuern die Interpreter. Sie haben eine festgelegte Syntax, die unten erklärt wird. Der Pfad des Konfigurationsfiles wird dem Interpreter in der Variable „file“ übergeben, das geschieht - in Abhängigkeit vom Ort des Aufrufes - folgendermaßen:

Aufruf	Pfadangabe	Beispiel
Aus einer HTML-Form	<input type=hidden name=file value=PFAD>	<input type=hidden name=file value=v/test.cfg> setzt die Variable file auf den Wert „v/test.cfg“ (relativ zum Pfad des Interpreters)
Aus einem Link	<a href=“INTERPRETER?file=PFAD“>	<a href=sF.pl?file=v/test.cfg> ist ein Link auf den Interpreter sF.pl (Abkürzung für „showForm“) mit der Konfigurationsdatei „v/test.cfg“.

Tab. 15: Interpreter - Parameterübergabe

In HTML-Forms wird der Interpreter im ACTION-Argument des FORM-Statements festgelegt. Beispiel: <form method=post action=showForm.pl> ... </form>

Die Optionen des Konfigurationsfiles haben den folgenden Aufbau: <OPTION> { ... } Wichtig ist, daß nach dem Ende einer Option immer eine Zeilenschaltung folgt, da alle Zeichen nach der geschwungenen Klammer ignoriert werden.

Folgende Optionen stehen zur Auswahl:

Option	Beschreibung
S	SQL-Statement in showForm und showList, über das Parameterarray wiederholtes SQL-Statement in makeManipulation
SH	Einzelnes SQL-Statement in makeManipulation. Mehrere SQL-Statements können durch Strichpunkte getrennt angegeben werden.
VF	Verweis auf eine Vorlagendatei
LE	Vorlage für einen Listeneintrag in showList.
HL	HTML-Text der Seitenüberschrift
TL	Text des Seitentitels (ausgegeben im Hauptfenster des Browsers)
H	Text, der vor einer Vorlage oder dem ersten Listeneintrag ausgegeben wird.
T	Text, der nach der Vorlage oder dem letzten Eintrag ausgegeben wird.
P	Definition von Parametern. Parameter werden in der Form PARAM=WERT definiert (Beispiel: anzahl=10). Mehrere Parameter können durch Leerzeichen getrennt definiert werden.
E0	Text, der ausgegeben wird, wenn eine Liste keine Elemente enthält oder eine Datenbankmanipulation keinen Satz betroffen hat.
EH	Text, der ausgegeben wird, wenn das SQL-Statement in SH einen Fehler verursacht hat.
EM	Text, der ausgegeben wird, wenn das SQL-Statement in S einen Fehler verursacht hat.
L	Sicherheitslevel des Scripts: 0 (unsicher), 1 (überprüft Login) oder 2 (überprüft Login und Autorisierung)

Tab. 16: Konfigurationsfile - Optionen

4.4 Variablen und Vorlagen

Vorlagen sind Dateien, die HTML-Text enthalten, wobei anstelle von konkreten Informationen Variablen stehen, die ggf. durch Datenbankwerte oder Benutzereingaben ersetzt werden. Als Variablen werden Strings der Form $<STRING> betrachtet. Die Interpreter versuchen nach dem Ausführen ihrer jeweiligen SQL-Statements, diese Variablen durch den Inhalt von gleichnamigen Perl-Hasheinträgen des assoziativen Arrays „%ph“ zu ersetzen. Der Inhalt der Vorlagenvariable $x würde beispielsweise durch den Inhalt der Perl-Variable $ph{„$x“} ersetzt werden.

Neben einfachen Variablen können auch Arrays definiert werden: $<ARRAY>-<NR> verweist auf den Wert des assoziativen Hash-Arrays „%pl“ an der Stelle {<ARRAY>}[<NR>]. So wird z. B. durch den String $haustier-5 auf die Variable $pl{„haustier“}[5] verwiesen. Arrays sind dort sinnvoll zu verwenden, wo nach Ausgabe einer Liste im nächsten Bearbeitungsschritt einen Datenbankmanipulation für jeden Listeneintrag durchzuführen ist (z. B. Lehrveranstaltungen anmelden). Die Anzahl der Wiederholungen wird von makeManipulation automatisch erkannt. Um in einer Liste die Nummer des aktuellen Listeneintrages auszugeben, kann die Pseudovariable $zeile verwendet werden. Beispiel (aus dem Konfigurationsfile von „LVA-Liste ausgeben“):

```
LE { <input type=checkbox name=lvnr-$zeile value=$2> $0:
$1<br> }
```

Hier wird das Array „lvnr“ definiert. Die Werte werden vom Benutzer eingegeben. Variablen, die nur aus Zahlen bestehen (wie $0, $1, usw.) werden durch den Inhalt der Spalte des aktuellen SQL-Statements ersetzt. Im obigen Beispiel sieht das SQL-Statement folgendermaßen aus:

```
S { select lvnr,name,lv# from lv
    where meldung_von<=sysdate and meldung_bis>=sysdate
}
```

Der Interpreter (showList) tut nun folgendes: erst wird das SQL-Statement ausgeführt und anschließend für jede Ergebniszeile die Parameter im String der LE-Option durch Ergebniswerte ersetzt ($0 durch lvnr, usw.). $zeile wird jeweils durch die Nummer des Datensatzes in der Ergebnismenge ersetzt. Anschließend wird die Zeile ausgegeben.

Variablen können ebenso in Vorlagen enthalten sein. Beispiel (Suchabfrage für Publikationen):

```
...
Publications since <input type=text name=dseit value="$0"
size=9 maxlength=9>
...
```

Hierbei wird der Parameter $0 durch den ersten Wert des Ergebnissatzes des zugehörigen SQL-Statements ersetzt. Der Interpreter ist logischerweise showForm.pl

(da das Ergebnis einer Einzelsatzabfrage in eine Vorlage ausgegeben wird). Schließlich können auch SQL-Statements Parameter enthalten:

```
SH { update publikation set titel='$titel',url='$url' where
pubnr=$pubnr }
```

Dabei wird $titel (von makeManipulation) durch den Wert der Perl-Variable $ph{„titel“} ersetzt, die zuvor mithilfe eines HTML-Formulars, welches ein Feld namens „titel“ enthielt, definiert wurde.

Variablen kann auch ein bestimmter Datentyp zugewiesen werden. Beginnt ein Variablenname mit „n“ (z. B. $nzahl), so handelt es sich um eine Nummer, beginnt er mit „d“, so um ein Oracle-Datum, ansonsten handelt es sich um ein Textfeld. Die Verwendung dieser Präfixe ist nur in HTML-Forms sinnvoll; die Datentypen werden von den Interpretern überprüft und ev. eine Fehlermeldung ausgegeben. Zu beachten ist ferner, daß das Präfix vor der Ersetzung weggestrichen wird. $nzahl wird also durch die Variable $ph{„zahl“} ersetzt.

Schließlich können in Vorlagen noch Inline-Statements definiert werden. Ein String der Form { <STRING> } wird an das Betriebsystem weitergegeben und dort ausgeführt. Die Ergebnisse werden in die Vorlage eingebunden. Auf diese Art können verschachtelte Abläufe realisiert werden. Auch aus Listeneinträgen können Inline-Statements aufgerufen werden; allerdings mit der Syntax [<STRING>].

4.5 Einsatzbeispiele

Als erstes Beispiel wird der Ablauf „Whiteboardeintrag ändern“ besprochen. Danach folgt als Beispiel für Inline-Statements „Lehrveranstaltungsinformationen ausgeben“.

Das Ändern von Whiteboardeinträgen läuft folgendermaßen ab:

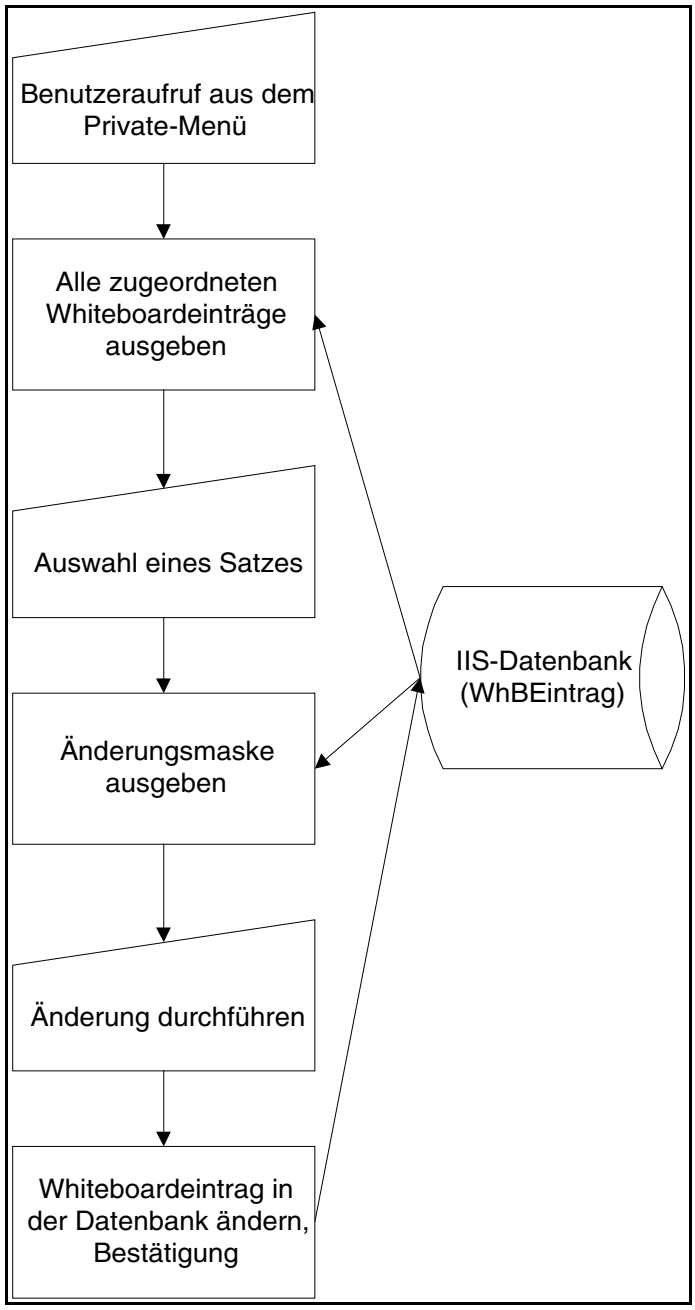

Abb. 27: Whiteboardeintrag ändern - Ablauf

Die Liste aller passenden Whiteboardeinträge wird mit showList und folgender Konfigurationsdatei ausgegeben:

```
S { select text,datum,bez,whbeintrag.rowid
    from whbeintrag,whbkategorie
    where whbeintrag.wknr=whbkategorie.wknr and pnr=$uid
    order by datum desc,bez
}
LE { <li> <a
     href="sF.pl?file=v/showWHBChg.cfg&uid=$uid&wid=$3">$0</a>
     ($1, $2)<br>
}
TL {Whiteboard - Eintrag &aumlndern }
```

```
HL {Whiteboard - Eintrag &aumlndern }
H { <blockquote>Clicken Sie auf den Text, um einen Eintrag zu
bearbeiten.<p><hr width=30%><p><ul>
}
T { </ul></blockquote>}
E0 { <center><h3>Sie haben derzeit keine Eintr&aumlge im
Whiteboard!</h3></center>}
```

Wenn der Benutzer dem entsprechenden Link folgt, bekommt er, die mit showForm und der folgenden Konfigurationsdatei ausgegebene Änderungsmaske:

```
S { select text,dauer,url,email,whbeintrag.wknr,bez
    from whbeintrag,whbkategorie
    where whbeintrag.wknr=whbkategorie.wknr and
          whbeintrag.rowid='$wid'
}
VF { v/showWHBChg.html }
...
```

Die Vorlage sieht dabei folgendermaßen aus:

```
...
<form method=post action=mM.pl>
Eintrag <input type=text name=txt value="$0" size=40
maxlength=255 >

...

<input type=submit value="   Eintragen   ">
<input type=hidden name=wid value=$wid>

...

</form>
```

Am Schirm stellt sich das folgendermaßen dar:

Abb. 28: Whiteboardeintrag ändern - Screenshot

Nach dem Drücken von „Eintragen“ erfolgt mithilfe von makeManipulation und der folgenden Konfigurationsdatei die Änderung des Datensatzes:

```
SH { update whbeintrag set text='$txt', dauer=$xdauer,
        url='$url', email='$email', wknr=$wknr
     where rowid='$wid'
}
HL { &Aumlnderung erfolgreich! }
...
```

Detailinformationen über Lehrveranstaltungen werden mithilfe folgender Vorlage ausgegeben:

```
...
<table border=3>
...
```

```
<tr><td>Vorbesprechung</td><td>$21 ($22)</td></tr>
<tr><td>Leiter</td><td>{perl sL.pl file=v/listLVALeiter.cfg
lvnr=$24}</td></tr>
<tr><td>Mitwirkende Assistenten</td><td>{perl sL.pl
file=v/listLVAMit.cfg lvnr=$24}</td></tr>
<tr><td>Anmeldestatus</td><td>$16 von $15</td></tr>
...
</table>
...
```

Der String „perl sL.pl ...“ wird dabei als Inline-Statement interpretiert und in einer eigenen Shell ausgegeben.

Verweise auf weitere Beispiele für Konfigurationsfiles und Vorlagen finden sich im 5. Kapitel des Anhangs.

4.6 Debugging

Fehler können im wesentlichen nur an vier Stellen gemacht werden:

1. Angabe einer falschen Interpreterfunktion

Der falsche Interpreter wird versuchen, die Konfigurationsdatei bestmöglich zu interpretieren. In jedem Fall kann man an der Ausgabe den Fehler leicht erkennen.

2. Fehler in SQL-Statements

Bei Syntaxfehlern folgt ein HTTP-Server-Error, da die auszugebende Seite dann nicht mit einem korrekten Header beginnt. Andere Fehler werden durch die Fehlerstrings im Konfigurationsfile abgefangen.

3. HTML- und Tippfehler

Werden an der Ausgabe erkannt.

4. Fehlerhafte Syntax in Konfigurationsdateien

Die Konfigurationsdateien werden von den Interpretern bestmöglich interpretiert. Manche Fehler fallen auf, andere nicht. Bei unerklärlichen Fehlern empfiehlt es sich, die Syntax der Konfigurationsfiles zu überprüfen.

5. Weiterführende Überlegungen

5.1 Erweiterung des Institutsinformationssystemes

Als weiteres Nebenprodukt des IIS-Webinterfaces neben der Entwicklungsumgebung wurde ein neues, erweitertes und normalisiertes Datenmodell fürs IIS entworfen, das u. a. eine Komponente zum Verwalten von Projekten und ein neues Rechtesystem enthält. Außerdem wurden mit der IIS-Realisierung die Voraussetzungen geschaffen, um eine schrittweise Umsetzung dieses erweiterten Konzeptes zu ermöglichen. Das erweiterte Datenmodell (das zugehörige Datenbankschema findet sich im Anhang) sieht folgendermaßen aus:

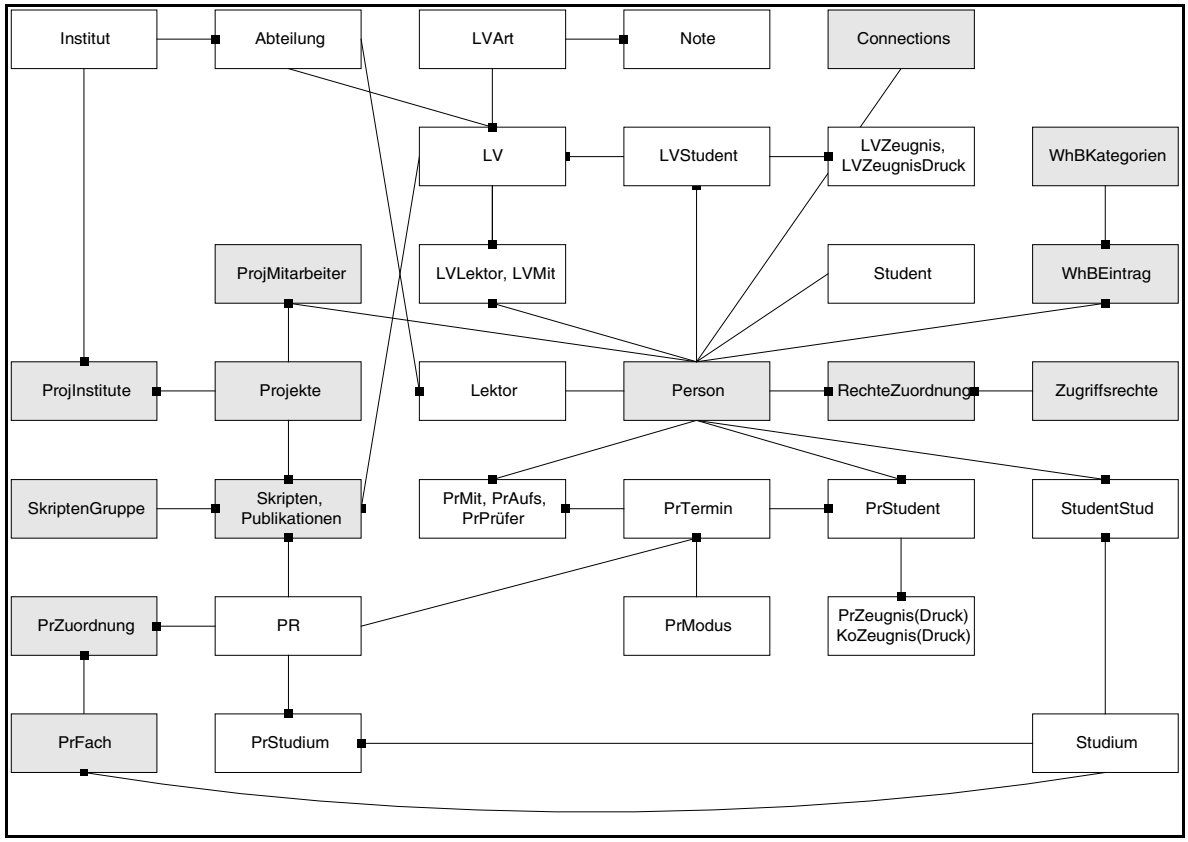

Abb. 29: Erweitertes Datenmodell

Desweiteren wurde ein erweiterter Taskplan ausgearbeitet, der sich ebenfalls im Anhang findet. Hier werden nur die wichtigsten Tasks angegeben (die Benutzergruppen entsprechen jenen in Kapitel 2.2):

Task	Benutzergruppe
Institute verwalten	M
Abteilungen verwalten	M
LV-Arten verwalten	M
Noten verwalten	M
Studenten verwalten	M

Lektoren verwalten	M
Personen verwalten	M
Rechte verwalten	M
Rechtezuordnung verwalten	M
Skriptengruppen verwalten	M
Studien verwalten	M
Prüfungsmodi verwalten	M
Prüfungen verwalten	M
Prüfungskategorien verwalten	M
Prüfungen Studien zuordnen/trennen	M
System konfigurieren	M
Loginstatus lesen	M
Drucker verwalten	M
Lehrveranstaltungen verwalten	L
Lehrveranstaltungen beurteilen	L
LVA-Mitwirkende verwalten	L
LVA-Zeugnisse ausdrucken	L
Skripten/Publikationen verwalten	L
Prüfungstermine verwalten	L
Prüfungsmitwirkende verwalten	L
Prüfungen beurteilen	L
Prüfungszeugnisse ausdrucken	L
Projekte verwalten	L
Projektmitwirkende verwalten	L
Projektinstitute verwalten	L
LVA-Anmeldestatus lesen	L
Prüfungs-Anmeldestatus lesen	L
LVA-Anmeldeliste generieren	L
Prüfungs-Anmeldeliste generieren	L
LVA-Ergebnisliste ausgeben	L
Prüfungs-Ergebnisliste ausgeben	L
Privates Menü aus Rechten generieren	E
Institutsinformationen lesen	A
Abteilungsinformationen lesen	A
Projektinformationen lesen	A
Studenteninformationen lesen	A
Skripteninformationen lesen	A
Studieninformationen lesen	A

Tab. 17: Erweiterter Taskplan (Überblick)

Ziel dieser IIS-Erweiterung ist es, den Benutzern mehr Informationen anzubieten und alle Datenwartungstätigkeiten über HTML-Forms ausführbar zu machen.

Die Implementierung der Datenbank sollte bereits unter Oracle 7 stattfinden. Unter Anwendung des vorgestellten Vorgehensmodelles sollte nach dem Design, der Freigabe und Evaluation einer Komponente dieselbe an den passenden Stellen des bestehenden IIS hinzugefügt werden. Umstellungen, die Änderungen bestehender

Komponenten erfordern (z. B. die Trennung von Personenstammdaten von Lektorendaten), müßten erst in einer Testumgebung entwickelt werden.

5.2 Vernetzung von Instituten

Das IIS würde sich, an mehreren Instituten verwendet, gut für Vernetzung von Komponenten eignen. So könnten beispielsweise mehrere, mit IIS ausgestattete Institute einer Universität ihre Studentenstammdaten austauschen, um den Studenten an allen Instituten die Anmeldung mit demselben Paßwort zu ermöglichen. Eine andere Anwendung wäre die Vernetzung von Instituten verschiedener Universitäten mit gleichem Forschungsauftrag, um z. B. eine gemeinsame Projektdatenbank aufzubauen. Schematisch stellt sich das folgendermaßen dar:

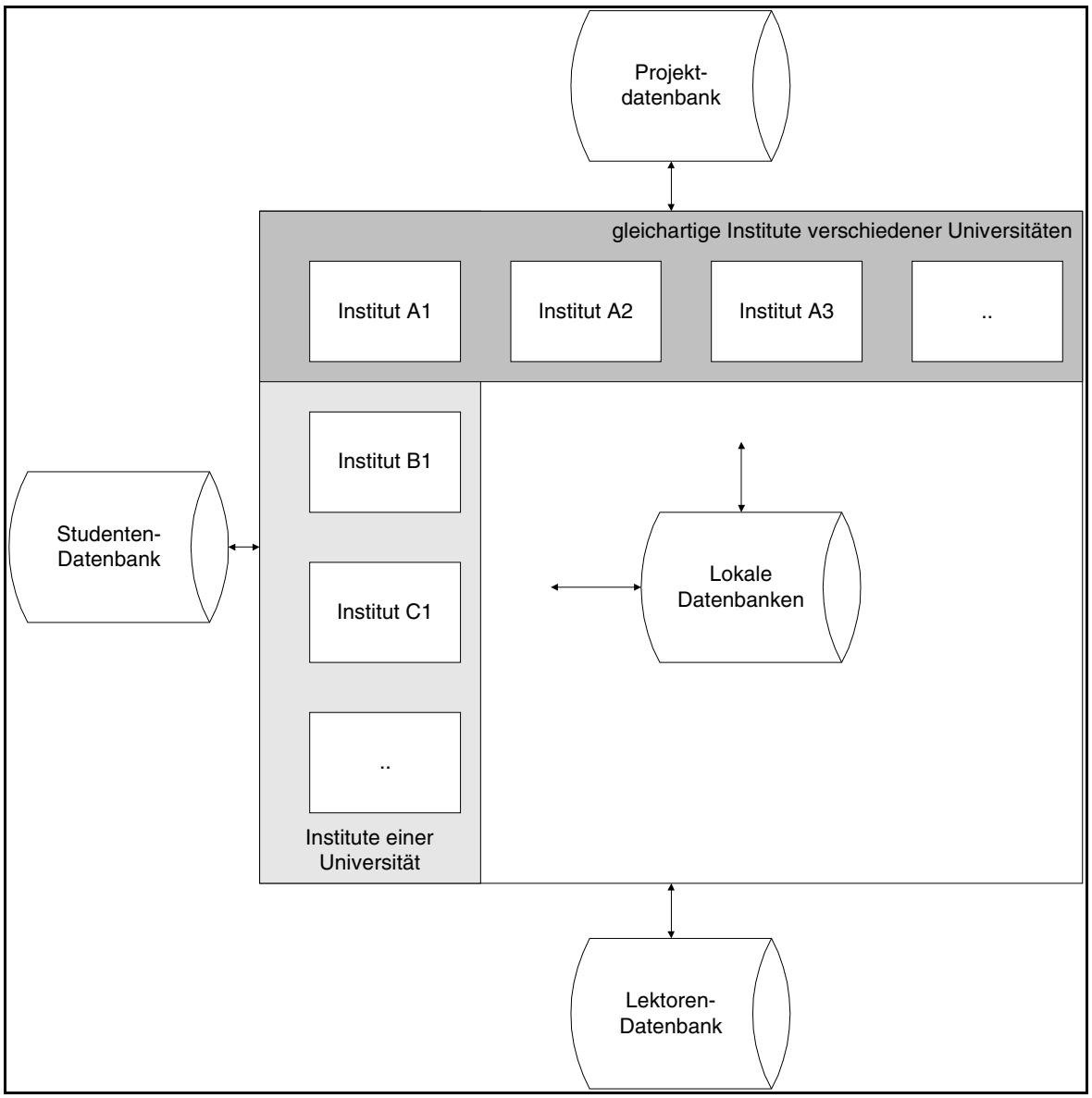

Abb. 30: Vernetzung von Instituten

Sowohl für die intra- als auch für die interuniversitäre Verknüpfung von Instituten interessant, wäre der Austausch von Lektorenstammdaten (Adreßdatenbank nach Fachgebieten, gemeinsame Publikationen, usw.).

Das Hauptproblem der Verknüpfung ist die verteilte Datenhaltung. Ein Lösungsvorschlag könnte sein, die gesamte Datenbank (mit externen Studenten- und Lektorendaten, usw.) an jedem Institut lokal zu halten und nur die Änderungsdaten in periodischen Abständen auszutauschen. Dies könnte geschehen, indem in der Datenbank alle Änderungen mitprotokolliert werden und diese Protokolle unter den Teilnehmern dieses „IIS-Verbundes“ ausgetauscht werden. Hier erscheint es sinnvoll, eine zentrale Stelle zu bilden, deren Aufgabe es ist, die Änderungsprotokolle zu sammeln, zu verbinden und an alle Teilnehmer weiterzuleiten. Außerdem könnte sich diese Stelle mit der Weiterentwicklung des Systems beschäftigen.

Der Nachteil dabei ist der steigende Aufwand für die Administration und außerdem gibt es erfahrungsgemäß stets Spannungen zwischen den Teilnehmern und der Zentralabteilung eines Verbundes. Allerdings ließe sich der Austausch der Änderungsprotokolle wohl relativ einfach automatisieren. Periodisch an den Teilnehmersystemen ablaufende Tasks würden die Protokolle aus der Datenbank herausnehmen und an einen permanent laufenden Demon auf einem als Zentralrechner vereinbarten Host senden, der die Aufgabe der Zusammenfassung und Weitergabe an die Teilnehmer übernähme. Das kommt vom Ablauf her tw. einem Publishing-Service nahe, wie er in Kapitel 5.3 besprochen wird.

Unter Umständen könnte es auch Sinn machen, das IIS mit Partnerorganisationen aus dem nicht-universitären Bereich zu verknüpfen (beispielsweise über Projektdaten, usw.). Da Firmen zumeist über eigene, viel hübschere Webserver verfügen, deren Datenhaltungskomponente oft nicht mit dem tollen Oberflächenlayout schritt hält (weshalb man sie als oberflächlich bezeichnen könnte?), hängt die Integration stark von der vorangegangenen Installation einer Datenbank ab, was einen großen Ressourcenaufwand darstellt.

Schließlich sind Datenschutz- und Schulungsaspekte zu berücksichtigen. Ersterer ist durch die Weitergabe von Personendaten in frage gestellt. Allerdings werden diese Personendaten auf anderen Servern des IIS-Verbundes auch nicht anderes präsentiert als auf dem eigenen Server. Es stehen also dieselben Informationen der Öffentlichkeit zur Verfügung wie im lokalen System.

Schulung wurde in der IIS-Entwickung vernachlässigt. Einerseits weil es unmöglich ist, die betroffenen Studenten zu schulen (diesbezügliche Versuche würden in einem so lernunwilligen Milieu kläglich scheitern :-) und andererseits, weil sich bei den sonstigen betroffenen Personengruppen (Lektoren, usw.) um Personen handelt, die erstens wenig Zeit und zweitens die Fähigkeit haben, sich die nötige Kenntnisse selbst zu erarbeiten. Wo es nötig war, wurden in den IIS-Seiten direkt Kommentare eingefügt. Die Einführung des IIS an anderen Instituten würde allerdings die Schulungsfrage neu stellen. Glücklicherweise ist sie mehr als unwahrscheinlich.

5.3 Andere Aufgabenstellungen

Im folgenden wird kurz die Eignung der IIS-Entwicklungsumgebung für drei andere Aufgabenstellungen besprochen: ein Bibliothekssystem für Universitätsinstitute, einen Server für Produktpräsentationen und einen Publishing-Server.

Ein Bibliothekssystem eines Universitätsinstitutes hat die folgenden Benutzergruppen:

Nr.	Benutzergruppe	Anmerkung
1	Bibliothekare (B)	Verwalten Bücher, Entlehnungen, usw.
2	Lektoren (L)	Entlehnen Bücher, führen Recherchen durch, usw.
3	Studenten (S)	Lesen Bücher am Institut, usw.

Tab. 18: Benutzergruppen eines Bibliothekssystemes

Folgende Aufgaben hat ein einfaches Bibliothekssystem zu unterstützen:

Nr.	Task	Benutzergruppe
1	Medien (Bücher, Sammelbände, usw.) verwalten	B
2	Benutzer verwalten	B
3	Bücher suchen (OPAC)	L, S
4	Bücher intern entlehnen	S
5	Bücher extern entlehnen	L
6	Entlehner mahnen	B
7	Bücherrückgabe	L

Tab. 19: Tasks eines Bibliothekssystemes

Der Task „Medien verwalten“ besteht beispielsweise aus den folgenden Subtasks: Medien katalogisieren, Einträge ändern und Medien ausscheiden. Jeder Subtask stellt einen Ablauf dar, der wiederum in zwei oder mehr Schritten mithilfe der Interpreter-Scripts zu realisieren ist.

Stellt das Bibliothekssystem eine Erweiterung des IIS dar, so läßt es sich folgendermaßen zum System hinzufügen:

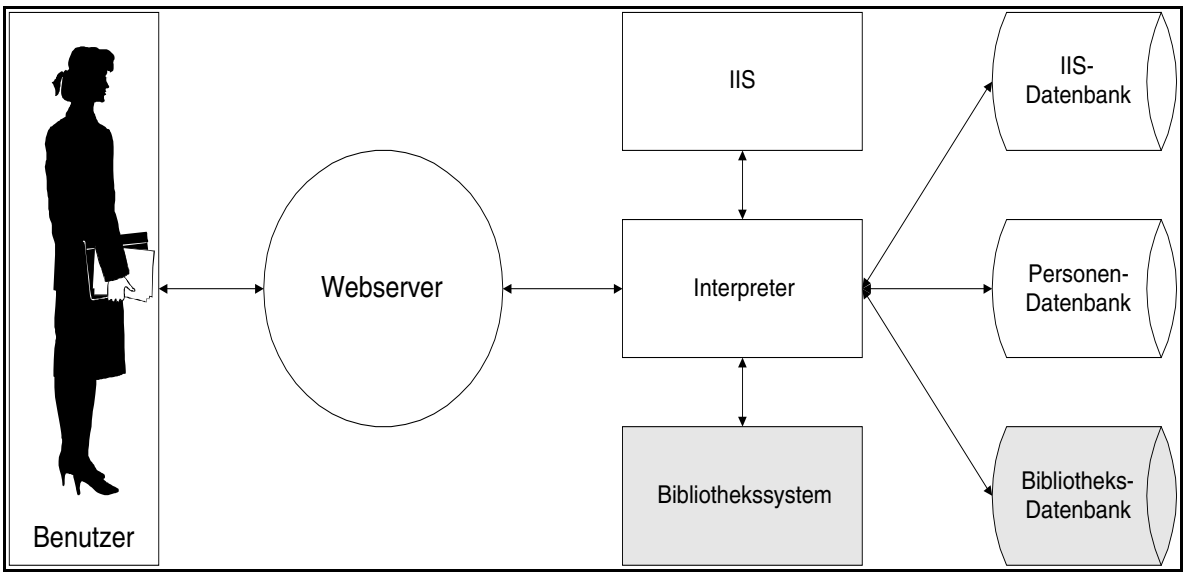

Abb. 31: Erweiterung des IIS um ein Bibliothekssystem

Für die bibliotheksspezifischen Daten (Medien und ihre Subtypen, Entlehnungen, ...) wären eigene Tabellen anzulegen; die Benutzerdaten könnten aber der Personen-Datenbank des IIS entnommen werden.

Insgesamt ergeben sich acht bis zehn zu realisierende Abläufe, zusammen mit der Erweiterung der Datenbank ergibt sich eine Entwicklungszeit für einen einzelnen Systementwickler von wenigen Wochen. Die Eignung der Entwicklungsumgebung für die Aufgabenstellung ist als „gut“ einzuschätzen.

Die Präsentation von Produkten im Internet (deren Erfolg generell eher zweifelhaft ist) basiert auf marketingorientierten Aspekten. Was verkauft werden soll, muß gut aussehen, ein einprägsames, eigenes Styling haben und in einem passenden Ambiente dargeboten werden. Das starre, unambitionierte Layout des IIS ist dafür, auch wenn man es mit poppigen Grafiken versieht, eher ungeeignet. Da bei Produktpräsentationen in der Regel auch nur wenige Daten anfallen, ist eine Datenbank im Hintergrund auch kaum notwendig. Anders sähe es etwa bei einem Anbieter von elektronischen Kleinteilen aus, dem es - wie den IIS-Verantwortlichen - eher um die Minimierung des Wartungsaufwandes als um geschniegelte Präsentation ginge.

Schließlich wird die Frage untersucht, inwieweit sich die IIS-Entwicklungsumgebung für ein Online-Publishing-Service (OPS) eignen würde. Ein solches System hat zwei Benutzergruppen: Autoren (Sender) und Leser (Empfänger), wobei eine Person abwechselnd in beide Rollen schlüpfen kann. Das System könnte etwa die folgenden Aufgaben wahrnehmen:

Nr.	Task	Benutzergruppe
1	Benutzer verwalten	System
2	Interessengruppen verwalten	A
3	Dokumente verwalten	A
4	Dokumente konvertieren	L
5	Dokumente verteilen	A
6	Dokumente online darstellen	A
7	Dokumente recherchieren	L

Tab. 20: Tasks eines Publishing-Services

Autoren versenden ihre Publikationen per eMail an das Publishing-Service mit einer Spezifikation, was damit geschehen soll. Möglichkeiten wären etwa die Weiterversendung an die Leser verschiedener Interessengruppen (Leser deklarieren ihre Interessen bei der Anmeldung selbst), die Publizierung als HTML-Text am Publishing-Server oder die Ablage in der Datenbank, wo Leser nach der Publikation recherchieren können und sich den Text ev. (ganz oder auszugsweise) vom System (in einem Format ihrer Wahl) schicken lassen können.

Mithilfe der IIS-Entwicklungsumgebung könnten die Verwaltungstasks und die Recherche realisiert werden. Durch Entwicklung einer Interpreter-Funktion, die Mailboxen auswertet und das Ergebnis in der Datenbank ablegt (abgeleitet von „makeManipulation“) ließen sich die übrigen Tasks realisieren.

Das Hauptproblem dieser Aufgabenstellung ist die Konvertierung von Dokumenten und dabei die Konvertierung von Grafikformaten. Ein Publishing-Service, das sich auf wenige Formate (wie Postscript, HTML, Rich Text Format, usw.) beschränkt, könnte von einer einzelnen Person aber binnen weniger Wochen realisiert werden und würde eine reizvolle Ergänzung für eine Institutshomepage darstellen.

5.4 Optimierung

Eine Optimierung und damit Performancesteigerung des IIS-Systems ließe sich auf die folgenden Arten erreichen:

Nr.	Ansatz
1	Erweiterte Seitengenerierung
2	Kürzere Abläufe
3	Weniger Sicherheitsabfragen
4	Weniger variable Entwicklungsumgebung
5	Höhere Priorität der CGI-Scripts

Tab. 21: Performance-Optimierung

Die Seitengenerierung (derzeit täglich für die Hauptseite, die Mitarbeiterliste, usw. verwendet) könnte weiter ausgedehnt werden. Beispielsweise könnten wochenweise für die Mitarbeiter die Detailinformations-Seiten generiert werden, wodurch die Links von der Mitarbeiterliste und den Lektoren-Listen der Prüfungen und Lehrveranstaltungen beschleunigt würden. Außerdem könnte man semesterweise die Prüfungs- und Lehrveranstaltungsseiten generieren. Es könnten alle Seiten generiert werden, die nicht mit Suchfunktionen erstellt werden. Die Nachteile wären immer dieselben: größerer Verbrauch an Datenspeicher und geringere Aktualität der Daten. Die bestehende Regelung ist ein Kompromiß zwischen diesen Nachteilen und einer schlechteren Performance.

Durch eine weitere Minimierung der Interaktionsschritte zwischen Benutzer und System ließe sich der Weg zu den relevanten Detailinformationen weiter abkürzen. Allerdings würde das Auffinden dieser Details ohne Suchfunktionen und Kategorisierungen viel schwieriger.

Sicherheitsabfragen kommen in jedem Ablauf mehrfach vor und verbrauchen einen großen Teil der Antwortzeit eines Interpreters. Durch Verzichten auf Sicherheitsabfragen, wo sie nicht unbedingt notwendig sind, ließe sich die Performance weiter steigern. Der Login eines Users könnte z. B. nur bei Datenmanipulationen (mit „makeManipulation“) abgefragt werden. Allerdings hätten Hacker dann die Möglichkeit, durch geschicktes Formulieren von Abfragen, geschützte Informationen zu lesen.

Würde man in der Entwicklungsumgebung auf mache Optionen verzichten (z. B. Inline-Statements, Kommandozeilenparameter, usw.), so ließen sich einige Programmabfragen vermeiden und die Performance steigern. Dadurch ergäbe sich aber eine Einschränkung der Funktionalität der Interpreter, so daß sich manche Operationen nicht mehr realisieren ließen.

Schließlich könnte der Systemadministrator den Interpreter-Tasks und der Datenbank höhere Priorität beim Scheduler einräumen oder das System exklusiv auf einem Rechner installieren. Zum höheren Kostenaufwand, den das Betreiben des Systems dann hätte, käme aber auch noch die Fähigkeit von Unix hinzu, einen die höhere Priorität nicht spüren zu lassen.

Zusammenfassend läßt sich feststellen, daß in allen fünf Punkten, in denen sich eine Optimierung des Systems erreichen ließe, Kompromisse zwischen höherer Performance und anderen Kriterien (wie Funktionalität, Datenschutz, usw.) eingegangen wurden, von denen ich glaube, daß sie vertretbar sind.

5.5 Weiterentwicklung der Entwicklungsumgebung

Vorausgesetzt man sieht in der IIS-Entwicklungsumgebung keinen programmierer-feindlichen Anachronismus und ist deshalb bereit, sie zu verwenden, so bieten sich die folgenden Möglichkeiten an, sie zu erweitern:

Nr.	Verbesserung
1	Intelligentere Variablenersetzung
2	Verschachtelte Inline-Statements
3	Mehr Fehleroptionen
4	Gemeinsames Editieren von Tabellen in 1:n-Relation

Tab. 22: Weiterentwicklung der Entwicklungsumgebung

Die Variablenersetzung besteht derzeit darin, daß alle Strings der Form $<STRING> durch - falls vorhanden - den Variablenwert von $ph{„STRING“} ersetzt werden. Intelligenter wäre eine Lösung, die einerseits den Variableninhalt mitsamt seiner Umgebung und andererseits nur bedingt ausgibt.

Beispiel: um eine Liste von Namen und Geburtsdaten der Form „$name ($gebdat)“ auszugeben, sollten die Klammern rund um $gebdat nur ausgegeben werden, wenn $gebdat ein richtiges Datum enthält.

Ein Lösungsvorschlag für dieses Problem könnte die folgende Variablensyntax sein:

```
„$VAR:/REGEX1/AUSGABE1:/REGEX2/AUSGABE2:...“
```

Stimmt der Inhalt von $VAR mit der Regular Expression REGEX1 überein, so wird AUSGABE1 ausgegeben, usw. Es wird immer nur die erste passende Option ausgegeben. Beispiel:

```
„$name $gebdat:/\d\d\-\w\w\w\-\d\d/($gebdat):/.*/“
```

Hierbei wird $gebdat nur (mit Klammern) ausgegeben, wenn es einem Oracle-Datum entspricht. In allen anderen Fällen wird nichts ausgegeben. Auf Systemebene ließe sich die Variablenersetzung durch ein eigenes assoziatives Array realisieren, das für jeden Variablennamen die verschiedenen regulären Ausdrücke und Ausgaben enthält.

Derzeit können Inline-Statements (mit „{}“ in Vorlagen bzw. mit „[]“ in Listenvorlagen eingeklammerter Text, der in einer eigenen Shell ausgeführt wird) nicht verschachtelt werden. Zwar kann eine Vorlage mehrere Inlines enthalten, aber niemals ineinander verkettet. Durch eine intelligentere Parameterprüfung („readParams“ in IIS.pm), die normale Kommandozeilenparameter von Inline-Statements unterscheiden kann, könnte diese Funktion realisiert werden. Zu beachten ist dabei, daß Verschachtelungen nur beim Inline-Aufruf von Interpreter-Funktionen relevant sind; Systemfunktionen können natürlich nichts damit anfangen.

Weiters könnten für die verschiedenen SQL-Fehler mehr Fehleroptionen als bisher (E0, EH und EM in Konfigurationsfiles) definiert werden. Derzeit werden alle Datenbankfehler, entsprechend ihrer Herkunft, durch die Fehlertexte in EH und EM abgefangen. Die Behandlung von leeren Ergebnismengen mit E0 ist eigentlich keine Fehlerabfrage. Das Problem bei mehr Fehleroptionen ist, daß die Datenbankschnittstelle (DBI.pm) ihrerseits nur die Tatsache eines Fehlers zurückgibt und nicht, um welchen es sich exakt handelt. Dies kann mit einer eigenen Fehlerfunktion abgefragt werden.

Die erweiterte Fehlerbehandlung ließe sich in den Konfigurationsdateien folgendermaßen realisieren:

```
E { SQLCODE:TEXT ...}
```

So ließe sich für jeden SQL-Fehlercode und jede Seite eigene Fehlermeldungen realisieren. Die Einträge EH und EM könnten dann als Defaultwerte für alle nicht abgefragten Werte verwenden. Beispiel:

```
E { -1411:Ende des Sets erreicht }
```

In den Interpretern könnte diese Zusatzfunktion wieder durch ein eigenes Array, das für den Fehlerwert den Fehlertext enthält, realisiert werden.

Schließlich wäre in manchen Situationen eine Funktion praktisch, die das gemeinsame Editieren von Tabellen, die zueinander in 1:n-Relation stehen, erlaubt. So könnten beispielsweise Publikationen und ihre Autoren gemeinsam editiert werden. Erreicht werden könnte das durch eine den Interpretern vorgeschaltete Funktion, die das Seitenausgabefenster des Browsers horizontal in zwei Frames teilt, im oberen die Ausgabe des Interpreters für die erste Tabelle und im unteren die Ausgabe des Interpreters für die zweite Tabelle ausgibt. Dieser Funktion müßten (als Link-Parameter) die Namen und die Konfigurationsdateien der beiden Interpreter mitgeteilt werden. Außerdem müßte dem zweiten Interpreter der Schlüssel des im oberen Fenster bearbeiteten Datensatzes bekannt sein.

Nach dem Abschluß der Datenmanipulation im oberen Fenster müßte (als Inlinefunktion) ein Script aufgerufen werden, das die Unterteilung des Screens wieder aufhebt.

Alle diese Erweiterungen würden die Funktionalität verbessern, das Erscheinungsbild der erstellten Seiten verschönern und die Benutzerfreundlichkeit erhöhen; sie würden aber auch die Performance verschlechtern.

6. Zusammenfassung

6.1 Modell

Ausgehend von dem Wunsch, mit möglichst geringem Aufwand in möglichst kurzer Zeit eine gute Weboberfläche für das Institutsinformationssystem des Institutes für Angewandte Informatik und Informationssysteme zu schaffen, wurde eine Methode entwickelt und an der Aufgabenstellung erprobt, mit deren Hilfe sich alle Arten von Webinformationssystemen auf sehr effiziente Weise entwickeln lassen. Das Entwicklungsmodell besteht im wesentlichen aus den folgenden Komponenten:

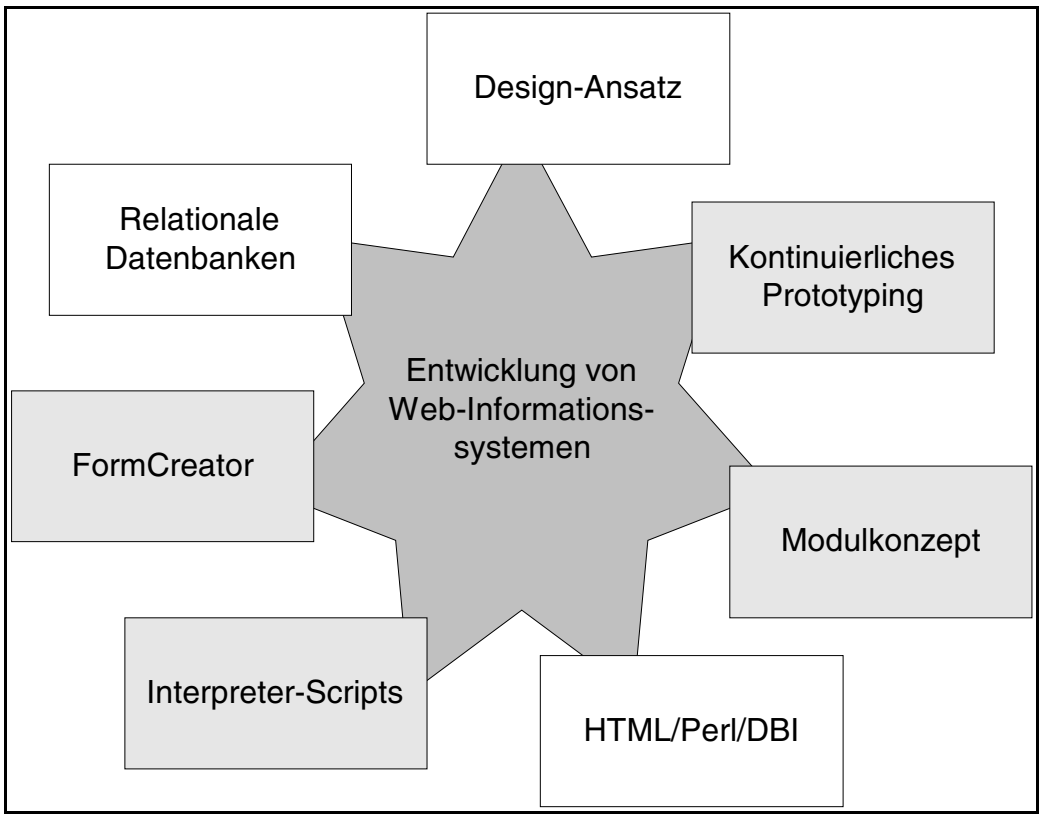

Abb. 32: Komponenten des Entwicklungsmodells

Voraussetzung für die Entwicklung ist die Verwendung einer relationalen Datenbank und der bewährten relationalen Datenmodellierungstechniken (EER-Diagramme, Normalisierung, usw.). Die Basis für den Entwicklungsprozeß ist das vom Designansatz in der Softwareentwicklung abgeleitete kontinuierliche Prototyping. Ausgehend von den Benutzerwünschen und ihren Aufgaben wird das System erst modelliert, anschließend die Entwicklungsbausteine entwickelt (Modulkonzept) und im iterativen Prototyping zur Anwendung verwoben, die schließlich nach den Wünschen der Benutzer gestaltet wird.

Die Kommunikation zwischen dem Benutzer und der Datenbank findet über Perl-CGI-Scripts statt; als Oberflächenbeschreibungssprache dient HTML. Dem Systementwickler stehen als Tools die durch Konfigurationsdateien steuerbaren

Interpreter und ein Form-Creator zur Verfügung, der aus SQL-Tabellenschemata Bildschirmmasken ableitet.

Bei Systemen, die nur die herkömmlichen Interpreter verwenden (Listen-, Maskenausgabe, Datenmanipulation) entfällt die Modulkonzept- und -realisierungsphase. Die Erweiterung von so erstellten Systemen beschränkt sich auf die Datenmodellierung, die Taskanalyse und das Prototyping.

6.2 Ergebnisse

Das Ziel dieser Diplomarbeit war die Erstellung eines Webinformationssystemes. Die Entwicklungsumgebung ist ein Nebenprodukt, dessen Vielseitigkeit in den weiterführenden Überlegungen hervorgehoben wurde. Das folgende Diagramm stellt die wesentlichen Ein- und Ausgaben des Entwicklungsprozeßes gegenüber:

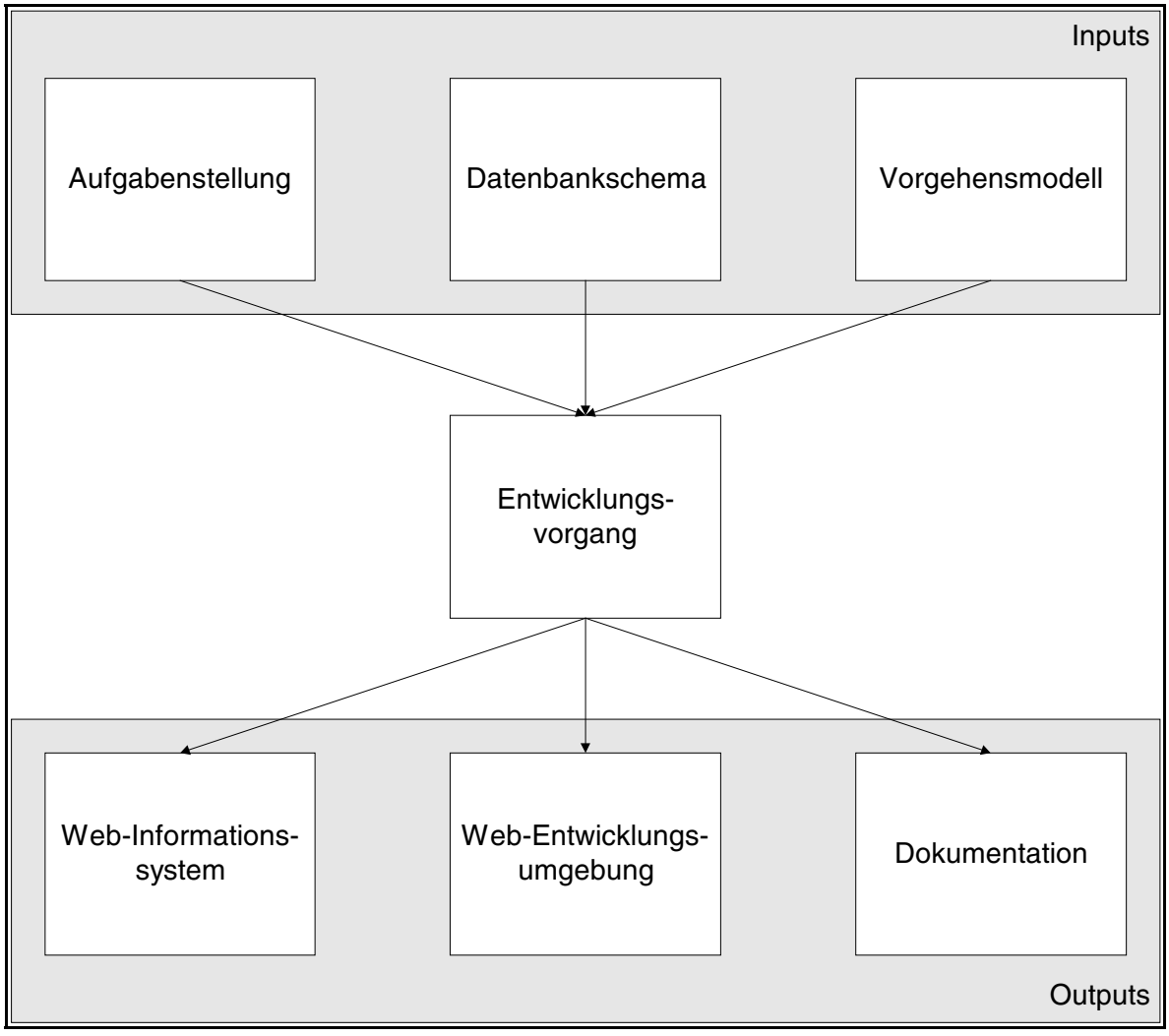

Abb. 33: Input/Output - Diagramm

Neben diesen Hauptergebnissen wurde auch noch ein Datenmodell des - bisher undokumentierten - IIS erstellt und es wurde ein Konzept für die Erweiterung des IIS durch zusätzliche Komponenten entwickelt. Schließlich wurde ein Vorgehensmodell

entwickelt, das sich eventuell auch für andere Entwicklungsprobleme (Server/Client-Programmierung , 4GL, usw.) eignet.

C

VERZEICHNISSE

1. Inhaltsverzeichnis

2. Abbildungsverzeichnis

3. Tabellenverzeichnis

4. Literaturverzeichnis

4.1 Bücher

[1] Developing user interfaces : ensuring usability through product & process
Deborah Hix ; H. Rex Hartson
New York NY, Wiley, 1993
ISBN 0-471-57813-4

[2] Datenbanken : Konzepte und Sprachen (1. Auflage)
Andreas Heuer ; Gunter Saake
Bonn, International Thomson Publ., 1995
ISBN 3-929821-31-1

[3] Informatik und Gesellschaft
Jürgen Friedrich ... (Hrsg.)
Heidelberg, Spektrum Akad. Verl., 1995
ISBN 3-86025-521-5

[4] Usability engineering (1st Edition)
Jakob Nielsen
Orlando Fla, Academic Press, 1993
ISBN 0-12-518405-0

[5] Graphical user interface design and evaluation (guide) : a practical process
David Redmond-Pyle ; Alan Moore
London, Prentice-Hall, 1995
ISBN 0-13-315193-X

[6] Einführung in Perl
Randal L. Schwartz
Köln, O'Reilly, 1996
ISBN 3-930673-08-8

[7] Perl 5 Desktop Reference
Johan Vromans
Köln, O'Reilly, 1996
ISBN 1-56592-187-9

[8] Oracle 7 - Server: SQL Language Reference Manual
Brian Linden
Oracle Inc., 1992
Part Nr. 778-70-1292

4.2 Skripten

[9] Vorlesung Datenmodellierung
WS 1996/97, Martin Hitz

[10] Vorlesung Datenbanksysteme
WS 96/97, Thomas Mück

[11] Vorlesung Sicherheitsmanagement
G. Pernul

[12] Übung Anwendungsprogrammierung PERL
SS 1996, E. Ellmer, C. Huemer

[13] Übung Anwendungsprogrammierung HTML/CGI
SS 1996, E. Ellmer, C. Huemer

[14] Methoden u. Werkzeuge für die Entw. von Benutzungsschnittstellen - Beispiel
H. Reiterer

[15] Methoden u. Werkzeuge für die Entw. von Benutzungsschnittstellen - Folien
H. Reiterer

[16] Vorlesung User Interface Design
M. Tscheligi

4.3 URLs

[17] xarch.tu-graz.ac.at/~caketin/perl/CGI.pm/index.html
Beschreibung des Perl CGI-Packages

[18] xarch.tu-graz.ac.at/~caketin/perl/manual/
Perl 5 - Online-Manual

[19] www.sandia.gov/sci_compute/html_ref.html
HTML 3.2 - Referenz

[20] www.hermetica.com/technologia/perl/DBI/doc/dbispec/dbidraft.html
Beschreibung des Perl DBI-Packages

[21] www.ani.univie.ac.at/~gabi/wwwconcept/index.html
IIS-Web-Konzept

D

ANHANG

1. Erweiterter Prototyping-Plan

Folgende Tasks sind für das erweiterte IIS zu implementieren. Die Prioritäten fassen Tasks zusammen und reihen sie (aufsteigend) nach ihrer Wichtigkeit. Die Benutzergruppen sind dieselben, wie sie in der Taskanalyse verwendet wurden.

Task / Subtasks	Priorität	Benutzergruppe
Institute hinzufügen - Leere Form ausgeben - Manipulation durchführen	2	M
Institute ändern - Institutsliste ausgeben - Gefüllte Form ausgeben - Manipulation durchführen	3	M
Institute löschen - Institutsliste ausgeben - Gewählte Institute löschen	2	M
Abteilungen hinzufügen - Leere Form ausgeben - Manipulation durchführen	2	M
Abteilungen ändern - Als Liste ausgeben - Gefüllte Form ausgeben - Manipulation durchführen	3	M
Abteilungen löschen - Als Liste ausgeben - Gewählte Einträge löschen	2	M
LV-Arten hinzufügen - Leere Form ausgeben - Manipulation durchführen	2	M
LV-Arten ändern - Als Liste ausgeben - Gefüllte Form ausgeben - Manipulation durchführen	3	M
LV-Arten löschen - Als Liste ausgeben - Gewählte Einträge löschen	2	M
Noten hinzufügen - Leere Form ausgeben - Manipulation durchführen	2	M
Noten ändern - Als Liste ausgeben - Gefüllte Form ausgeben - Manipulation durchführen	3	M
Noten löschen - Als Liste ausgeben - Gewählte Einträge löschen	2	M

Studenten hinzufügen - Leere Form ausgeben - Manipulation durchführen	0	M
Studentenstammdaten ändern - Matrikelnummer abfragen - Gefüllte Form ausgeben - Manipulation durchführen	1	M
Studenten löschen - Matrikelnummer abfragen - Sicherheitsabfrage - Gewählte Einträge löschen	1	M
Lektoren hinzufügen - Leere Form ausgeben - Manipulation durchführen	0	M
Lektorenstammdaten ändern - Als Liste ausgeben - Gefüllte Form ausgeben - Manipulation durchführen	1	M
Lektoren löschen - Als Liste ausgeben - Gewählte Einträge löschen	1	M
Personen hinzufügen - Leere Form ausgeben - Manipulation durchführen	0	M
Personenstammdaten ändern - Als Liste ausgeben - Gefüllte Form ausgeben - Manipulation durchführen	1	M
Personen löschen - Als Liste ausgeben - Gewählte Einträge löschen	1	M
Rechte hinzufügen - Leere Form ausgeben - Manipulation durchführen	0	M
Rechte ändern - Als Liste ausgeben - Gefüllte Form ausgeben - Manipulation durchführen	3	M
Rechte löschen - Als Liste ausgeben - Gewählte Einträge löschen	1	M
Rechte an Personen vergeben - Person selektieren - Nicht vergebene Rechte als Liste ausgeben - Manipulation durchführen	0	M
Rechte von Personen zurücknehmen - Person selektieren - Vergebene Rechte als Liste ausgeben - Manipulation durchführen	1	M
Skriptengruppen hinzufügen	2	M

- Leere Form ausgeben - Manipulation durchführen		
Skriptengruppen ändern - Als Liste ausgeben - Gefüllte Form ausgeben - Manipulation durchführen	3	M
Skriptengruppen löschen - Als Liste ausgeben - Gewählte Einträge löschen	2	M
Studien hinzufügen - Leere Form ausgeben - Manipulation durchführen	2	M
Studien ändern - Als Liste ausgeben - Gefüllte Form ausgeben - Manipulation durchführen	3	M
Studien löschen - Als Liste ausgeben - Gewählte Einträge löschen	2	M
Prüfungsmodi hinzufügen - Leere Form ausgeben - Manipulation durchführen	2	M
Prüfungsmodi ändern - Als Liste ausgeben - Gefüllte Form ausgeben - Manipulation durchführen	3	M
Prüfungsmodi löschen - Als Liste ausgeben - Gewählte Einträge löschen	2	M
Prüfungen hinzufügen - Leere Form ausgeben - Manipulation durchführen	1	M
Prüfungen ändern - Als Liste ausgeben - Gefüllte Form ausgeben - Manipulation durchführen	3	M
Prüfungen löschen - Als Liste ausgeben - Gewählte Einträge löschen	1	M
Prüfungskategorien hinzufügen - Leere Form ausgeben - Manipulation durchführen	2	M
Prüfungskategorien ändern - Als Liste ausgeben - Gefüllte Form ausgeben - Manipulation durchführen	3	M
Prüfungskategorien löschen - Als Liste ausgeben - Gewählte Einträge löschen	2	M
Prüfungen Studien zuordnen	1	M

- Form ausgeben - Manipulation durchführen		
Prüfungen/Studien-Zuordnung aufheben - Zuordnungen als Liste ausgeben - Manipulation durchführen	1	M
System konfigurieren - Status als gefüllte Form ausgeben - Manipulation durchführen	3	M
Loginstatus lesen - Status als Liste ausgeben	3	M
Drucker hinzufügen - Leere Form ausgeben - Manipulation durchführen	2	M
Drucker ändern - Als Liste ausgeben - Gefüllte Form ausgeben - Manipulation durchführen	3	M
Drucker löschen - Als Liste ausgeben - Gewählte Einträge löschen	2	M
Lehrveranstaltungen hinzufügen - Leere Form ausgeben - Manipulation durchführen	0	L
Lehrveranstaltungen ändern - Als Liste ausgeben - Gefüllte Form ausgeben - Manipulation durchführen	3	L
Lehrveranstaltungen löschen - Als Liste ausgeben - Gewählte Einträge löschen	0	L
Lehrveranstaltungen beurteilen - Lehrveranstaltung auswählen - Angemeldete Studenten als Liste ausgeben - Manipulation durchführen	1	L
LVA-Mitwirkende hinzufügen - Leere Form ausgeben - Manipulation durchführen	0	L
LVA-Mitwirkende löschen - Als Liste ausgeben - Gewählte Einträge löschen	0	L
LVA-Zeugnisse ausdrucken - Lehrveranstaltung auswählen - Person auswählen - Zeugnis drucken	2	L
LVA-Skripten hinzufügen - Leere Form ausgeben - Manipulation durchführen	2	L
LVA-Skripten entfernen - Als Liste ausgeben - Gewählte Einträge löschen	2	L

Prüfungstermine hinzufügen - Leere Form ausgeben - Manipulation durchführen	1	L
Prüfungstermine ändern - Als Liste ausgeben - Gefüllte Form ausgeben - Manipulation durchführen	3	L
Prüfungstermine löschen - Als Liste ausgeben - Gewählte Einträge löschen	1	L
Prüfungsmitwirkende hinzufügen - Leere Form ausgeben - Manipulation durchführen	1	L
Prüfungsmitwirkende löschen - Als Liste ausgeben - Gewählte Einträge löschen	1	L
Prüfungen beurteilen - Prüfung auswählen - Angemeldete Studenten als Form ausgeben - Manipulation durchführen	2	L
Prüfungszeugnisse ausdrucken - Prüfung wählen - Angemeldete Studenten als Liste ausgeben - Zeugnis drucken	2	L
Prüfungsskripten hinzufügen - Leere Form ausgeben - Manipulation durchführen	2	L
Prüfungsskripten löschen - Als Liste ausgeben - Gewählte Einträge löschen	2	L
Projekte hinzufügen - Leere Form ausgeben - Manipulation durchführen	2	L
Projekte ändern - Als Liste ausgeben - Gefüllte Form ausgeben - Manipulation durchführen	3	L
Projekte löschen - Als Liste ausgeben - Gewählte Einträge löschen	2	L
Projektmitwirkende hinzufügen - Leere Form ausgeben - Manipulation durchführen	2	L
Projektmitwirkende löschen - Als Liste ausgeben - Gewählte Einträge löschen	2	L
Projektskripten hinzufügen - Leere Form ausgeben - Manipulation durchführen	2	L
Projektskripten löschen	2	L

- Als Liste ausgeben - Gewählte Einträge löschen		
Projektinstitute hinzufügen - Leere Form ausgeben - Manipulation durchführen	2	L
Projektinstitute löschen - Als Liste ausgeben - Gewählte Einträge löschen	2	L
LVA-Anmeldestatus lesen - Lehrveranstaltung auswählen - Status als Form ausgeben	0	L
Prüfungs-Anmeldestatus lesen - Prüfung auswählen - Status als Form ausgeben	1	L
LVA-Anmeldeliste generieren - Lehrveranstaltung auswählen - Personen-Liste ausdrucken	0	L
Prüfungs-Anmeldeliste generieren - Prüfung auswählen - Personen-Liste ausdrucken	1	L
LVA-Ergebnisliste ausgeben - Lehrveranstaltung ausgeben - Ergebnisliste ausdrucken	1	L
Prüfungs-Ergebnisliste ausgeben - Prüfung auswählen - Ergebnisliste ausdrucken	2	L
Menü generieren - Zugeordnete Rechte als Menü ausgeben	0	E
Institutsinformationen lesen - Als Liste ausgeben - Detail-Form ausgeben	3	A
Abteilungsinformationen lesen - Als Liste ausgeben - Detail-Form ausgeben	3	A
Projektinformationen lesen - Als Liste ausgeben - Detail-Form ausgeben	3	A
Studenteninformationen lesen - Als Liste ausgeben - Detail-Form ausgeben	3	A
Skripteninformationen lesen - Als Liste ausgeben	3	A
Studieninformationen lesen - Als Liste ausgeben - Detail-Form ausgeben	3	A

Tab. 23: Erweiterter Taskplan (mit Subtasks)

2. Erweitertes Datenbankschema

Das erweiterte IIS basiert auf den folgenden (normalisierten, reduzierten, ...) Tabellenschemata (im Sourcetree unter „/sql/neu/create.sql“ zu finden):

```
create      table       abteilung   (
            abt_nr      number(2)   not null,
            abt_name    char(80),
            inst_nr     number(4),
            str         char(60),
            plz         char(5),
            ort         char(40)
);
create      table       drucker     (
            adresse     char(40),
            name        char(20),
            drucker_nr  number(2)   not null
);
create      table       institut    (
            inst_nr     number(4)   not null,
            inst_name   char(80),
            inst_url    char(30)
);
create      table       lektor      (
            pnr         number(10)  not null,
            sozialnr    number(10),
            sprechtag   char(2),
            sprech_von  date,
            sprech_bis  date,
            pers_nr     number(5),
            inst_nr     number(4),
            drucker_nr  number(2),
            abt_nr      number(2)
);
create      table       lv          (
            aenderungsdatum         date,
            inst_nr     number(4),
            abt_nr      number(2),
            ort_2       char(20),
            pa          char(1),
            remun       char(1),
            lv#         number(10)  not null,
            url         char(70),
            lvnr        number(6),
            semester    char(2),
            studienjahr number(4),
            name        char(80),
            tag_1       char(2),
            tag_1_von   char(5),
            tag_1_bis   char(5),
            tag_2       char(2),
            tag_2_von   char(5),
```

```
            tag_2_bis   char(5),
            ort         char(20),
            meldung_von date,
            meldung_bis date,
            meldmax     number(4),
            gemeldet    number(4),
            lvart       char(2),
            stunden     number(4),
            lvtyp       number(4),
            info_1      char(255),
            info_2      char(255),
            vbinfo      char(40),
            vb          date
);
create      table       lvart       (
            lvart       char(2),
            lvbez       char(80)
);
create      table       lvlektor    (
            lv#         number(10)  not null,
            pnr         number(10)  not null
);
create      table       lvmit       (
            lv#         number(10)  not null,
            pnr         number(10)  not null
);
create      table       lvstudent   (
            lvstudent#  number(10)  not null,
            lv#         number(10)  not null,
            matnr       number(10),
            knr_1       number(3),
            wartel      number(4),
            note        number(1),
            beurteilung char(50),
            datum       date,
            zeugnis     number(2)   not null,
            seq         number(4),
            flag_nr     number(4),
            modus       number(1)
);
create      table       lvzeugnis   (
            lvstudent#  number(10)  not null,
            zeugnis     number(2)   not null,
            druckdatum  date,
            drucker_nr  number(2)
);
create      table       lvzeugnisdruck  (
            lvstudent#  number(10)  not null,
            zeugnis     number(2)   not null,
            druckdatum  date,
            drucker_nr  number(2)
);
```

```
create      table       note        (
            note        number(1),
            lvart       char(2),
            text        char(30),
            beurt_code  char(1)
);
create      table       pr          (
            prnr        number(6)   not null,
            name        char(80),
            info        char(152),
            titel       char(230),
            stunden     number(4),
            url         char(70)
);
create      table       praufsicht  (
            prtermin#   number(10)  not null,
            pnr         number(10)  not null
);
create      table       prmit       (
            prtermin#   number(10)  not null,
            pnr         number(10)  not null
);
create      table       profile     (
            semester    char(2),
            studienjahr number(4)
);
create      table       prpruefer   (
            prtermin#   number(10)  not null,
            pnr         number(10)  not null
);
create      table       prstudent   (
            prnr        number(6)   not null,
            matnr       number(10)  not null,
            note        number(1),
            info        char(255),
            platz       number(4),
            prtermin#   number(6),
            seq         number(4),
            zeugnis     number(2),
            prstudent#  number(10),
            flag_nr     number(4),
            beurt_code  char(1),
            modus       char(1),
            knr_1       number(3)
);
create      table       prstudium   (
            prnr        number(6),
            knr         number(3)
);
create      table       prtermin    (
            prtermin#   number(6)   not null,
            prnr        number(6),
```

```
            dppnr       number(6),
            lv#         number(10),
            datum       date,
            beginn      char(5),
            dauer       char(5),
            ort         char(50),
            termin      number(1),
            meldebeginn date,
            meldeende   date,
            anmeldeende date,
            gemeldet    number(4),
            meldmax     number(4),
            inst_nr     number(4),
            abt_nr      number(2),
            info        char(76),
            listkopf    char(76),
            listinfo    char(200),
            modus       char(1),
            aenderungsdatum         date
);
create      table       pruefmodus  (
            modus       char(1),
            text        char(60)
);
create      table       przeugnis   (
            prstudent#  number(10)  not null,
            zeugnis     number(2)   not null,
            druckdatum  date,
            drucker_nr  number(2)
);
create      table       przeugnisdruck  (
            prstudent#  number(10)  not null,
            zeugnis     number(2)   not null,
            druckdatum  date,
            drucker_nr  number(2)
);
create      table       report      (
            srw_report_name         char(80),
            name        char(100),
            typ         char(1)
);
create      table       student     (
            pnr         number(10)  not null,
            matnr       number(10)
);
create      table       studentstud (
            matnr       number(10)  not null,
            kbu_a       char(1),
            knr_1       number(3),
            knr_2       number(3),
            knr_3       number(3),
            kbu_e       char(1)
```

```
);
create      table       studium     (
            knr         number(3),
            name        char(76),
            url         char(70)
);



create      table       capability  (
            cap         number(12)  not null
);
create      table       connection  (
            pnr         number(10)  not null,
            status      number(1),
            host        char(40)
);
create      table       projekt     (
            projnr      number(6)   not null,
            bez         char(80),
            url         char(70),
            beginn      date,
            dauer       number(4)
);
create      table       projmitarbeiter (
            projnr      number(10)  not null,
            pnr         number(10)  not null
);
create      table       projinstitut    (
            projnr      number(10)  not null,
            instnr      number(4)   not null
);
create  table   prfach  (
        fachnr  number(4)       not     null,
        studnr  number(3),
        bez     char(76),
        url     char(70)
);
create  table   przuordnung     {
        prnr    number(6),
        fachnr  number(4)
};
create      table       skripten    (
            sgnr        number(4),
            linknr      number(10),
            autor       number(10),
            bez         char(80),
            url         char(70)
);
create      table       skriptengruppe  (
            sgnr        number(4)   not null,
            bez         char(76)
```

```
);
create      table       person      (
            pnr         number(10)  not null,
            vorname     char(30),
            nachname    char(80),
            titel       char(20),
            password    char(15),
            geburtstag  date,
            geschlecht  char(1),
            email       char(40),
            str1        char(60),
            plz1        char(8),
            ort1        char(40),
            tel1        char(20),
            str2        char(60),
            plz2        char(8),
            ort2        char(40),
            tel2        char(20),
            url         char(80)
);
create      table       whbkategorie    (
            wknr        number(4)   not null,
            bez         char(76)
);
create      table       whbeintrag  (
            wknr        number(4),
            text        char(255),
            pnr         number(10),
            datum       date,
            dauer       number(4),
            url         char(70),
            email       char(40),
            adressat    number(1)
);
create      table       rechtezuordnung (
            pnr         number(10)  not null,
            rechtnr     number(4)   not null
);
create      table       zugriffsrecht   (
            rechtnr     number(4)   not null,
            action      char(80),
            bez         char(60),
            art         number(1)
);
create      table       publikation (
            pubnr       number(6),
            titel       char(80),
            url         char(80)
);
create      table       pubautor    (
            pubnr       number(6),
            lektnr      number(10),
```

```
            name        char(50)
);
```

3. Systemfunktionen

3.1 Masken ausgeben („showForm.pl“)

```
#!/usr/local/gnu/bin/perl

# showForm.pl: Form ausgeben
# hMe; 1997-07-30 bis 1997-08-06; ausgetestet: 1997-08-20

use CGI;          # HTML-Ausgabefunktionen
use DBI;          # Oracle-Zugriffsfunktionen
require IIS;      # IIS-Funktionen

# 1. Parameter lesen

&readParams;

# 2. Konfigurationsfile auswerten (O)

&checkConfig($ph{"file"});

# 3. Datenbankanmeldung (O)

if ($mainSQL ne "") {
   &logonDB;

   # 4. Securitytest (O)

   &checkSecurity($ph{"uid"},$ph{"file"});

   # 5. SQL-Head durchfuehren (O)

   $sth=$dbh->prepare(&substParams($mainSQL));
   $rc=$sth->execute;
   if ($rc!=0) {
      &logoutDB;

      if ($errorHead ne "") {
         printError("DB-Fehler: $rc",$errorHead);
      } else {
         printError("DB-Fehler: $rc","Fehler beim Zugriff auf
            die Datenbank.");
      }
   }

   # 6. Form ausgeben
```

```
   $col=0;
   my @erg=$sth->fetchrow_array;

   foreach $e (@erg) {
      $ph{"$col"}=$e;
      $col++;
   }

   $sth->finish;
}

&printHead($formTitle,$formHead);
&printForm($formFile);
&printTail;

# 7. Datenbankabmeldung (O)

if ($mainSQL ne "") {
   &logoutDB;
}
```

3.2 Listen ausgeben („showList.pl“)

```
#!/usr/local/gnu/bin/perl

# showList.pl: SQL-Ergebnistabelle als Liste mit Head & Tail
               ausgeben
# hMe; 1997-07-30 bis 1997-08-06

use CGI;          # HTML-Ausgabefunktionen
use DBI;          # Oracle-Zugriffsfunktionen
require IIS;      # IIS-Funktionen

# 1. Parameter lesen

&readParams;

# 2. Konfigurationsfile auswerten (O)

&checkConfig($ph{"file"});

# 3. Datenbankanmeldung (O)

$dbh=&logonDB;

# 4. Securitytest (O)

&checkSecurity($ph{"uid"},$ph{"file"});

# 5. Liste ausgeben
```

```
$sth=$dbh->prepare(&substParams($mainSQL));
if ($sth->execute!=0) {
   &logoutDB;

   if ($errorMain ne "") {
      &printError("Datenbank-Lesefehler",$errorMain);
   } else {
      &printError("Datenbank-Lesefehler","Fehler beim Lesen
         der Datenbank.");
   }
}

if ($formHead ne "") {                          # Kopftext
ausgeben
   &printHead($formTitle,$formHead);
   print &substParams($preText);
}

$anzSaetze=0;
$element=&substParams($element);    # Globale Parameter
ersetzen

while(@erg=$sth->fetchrow_array) {
   $anzSaetze++;

   $col=0;
   $zeile=&substElem($element,"zeile",$anzSaetze);

   foreach $e (@erg) {                 # SQL-Parameter ersetzen
      $zeile=&substElem($zeile,"$col",$e);
      $col++;
   }
   printLikeForm($zeile);
}

$sth->finish;

if ($anzSaetze==0) {
   if ($errorNull ne "") {
      print $errorNull;
   } else {
      print "Es wurde kein passender Eintrag in der Datenbank
         gefunden."
   }
} else {
   print &substParams($postText);
}

if ($formHead ne "") {                  # Fusstext ausgeben
   &printTail;
}
```

```
# 6. Datenbankabmeldung (O)

&logoutDB;
```

3.3 Datenbankmanipulationen durchführen („makeManipulation.pl“)

```
#!/usr/local/gnu/bin/perl

# makeManipulation.pl: Datenbankaenderung durchfuehren
# hMe; 1997-07-30 bis 1997-08-07; ausgetestet: 1997-08-21

use CGI;          # HTML-Ausgabefunktionen
use DBI;          # Oracle-Zugriffsfunktionen
require IIS;      # IIS-Funktionen

# 1. Parameter lesen

&readParams;

# 2. Konfigurationsfile auswerten (O)

&checkConfig($ph{"file"});

# 3. Datenbankanmeldung (O)

&logonDB;

# 4. Securitytest (O)

&checkSecurity($ph{"uid"},$ph{"file"});

# 5. SQL-Head durchfuehren (O)

$dbh->commit;
if ($headSQL ne "") {
   foreach $e (split(/;/,&substParams($headSQL))) {
      $anz=$dbh->do($e) || &stopOnHeadError;
   }
}

# 6. Manipulation durchfuehren

if ($mainSQL ne "") {
   $mainSQL=&substParams($mainSQL);

   $x=0;
   while($x<$maxPl) {
      foreach $e (split(/;/,&substParams($mainSQL,$x))) {
         $anz=$dbh->do($e) || &stopOnMainError;
      }
      $x++;
```

```
   }
}

$dbh->commit;

# Ausgabe

if ($formHead ne "") {
   &printHead($formTitle,$formHead);

   if ($anz==0 && $errorNull ne "") {
      print &substParams($errorNull);
   } else {
      print &substParams($preText.$postText);
   }

   if ($formFile ne "") {
      &printForm($formFile);
   }

   &printTail;
}

# 7. Datenbankabmeldung (O)

&logoutDB;


# Funktionenteil


# Nach Head-Fehler beenden

sub stopOnHeadError {
   $dbh->rollback;
   &logoutDB;

   if ($errorHead eq "") {
      &printError("DB-Fehler",
         "Fehler beim Zugriff auf die Datenbank!");
   } else {
      &printError("DB-Fehler",$errorHead);
  }
}

# Nach Main-Error beenden

sub stopOnMainError {
   $dbh->rollback;
   &logoutDB;

   if ($errorHead eq "") {
```

```
      &printError("DB-Fehler",
          "Fehler beim Zugriff auf die Datenbank!");
   } else {
      &printError("DB-Fehler",$errorHead);
   }
}
```

3.4 Wichtige Bibliotheksfunktionen aus „IIS.pm“

Ausgabe einer Form:

```
sub printForm {
   open(FD,$_[0]) || printError("Datei nicht gefunden",
       "Datei $_[0] konnte nicht ge&oumlffnet werden.");

   my $status=0;               # Normale Ausgabe als Default
   my $zeile="";
   my $erg="";

   while(<FD>) {
      $_=&substParams($_);     # Parameter ersetzen

      foreach $e (split(/\s*[\{\}]\s*/)) {

         if ($status==0) {          # Ausgabe
            print $e;
            unless ($e=~/\n$/) {    # Noch Elemente vorhanden
               $status=1;
            }
         } else {                   # Inline-Funktion
            $zeile=$zeile." ".$e;
            unless ($e=~/\n$/) {    # Einzeiliges Statement
               $erg=`$zeile`;
               print "$erg";

               $status=0;
               $zeile="";
            } else {
               chop $zeile;
            }
         }
      }
   }
}
```

Lesen der Parameter, die mithilfe der verschiedenen Übergabemöglichkeiten übergeben wurden:

```
sub readParams {
   $c=new CGI;
```

```
# 0. Systemvariablen setzen

$loginStatus=0;
$ph{"hn"}=$c->remote_host;

# 1. Kommandozeile

foreach $e (@ARGV) {
   ($n,$v)=split(/\=/,$e);

   unless ($v=~/^null$/i) {
      $ph{&cgi2ascii($n)}=&cgi2ascii($v);
   }
}

# 2. CGI-Parameter mit GET-Methode

$qs=`echo \$QUERY_STRING`;
unless ($qs=~/^\s*$/) {
   foreach $e (split(/&/,$qs)) {
      ($n,$v)=split(/\=/,$e);

      unless ($v=~/^null$/i || $ph{$n} ne "") {
         $ph{$n}=&cgi2ascii($v);
      }
   }

} else {

   # 3. CGI-Parameter mit POST-Methode

   $maxPl=0;     # Anzahl der zu bearbeitenden Zeilen

   foreach $e ($c->param) {
      if ($e=~/\w+\-\d+$/) {
         ($n,$i)=split(/\-/,$e);
         unless ($v=~/^null$/i) {
            if ($n=~/^[dn]/) {
               $n=~s/^(.)//;
               &syntaxCheck($1,$c->param($e));    # Syntax
                  ueberpruefen
            }

            $pl{$n}[$maxPl{$n}]=&cgi2ascii($c->param($e));
            $maxPl{$n}++;
            if ($maxPl<$maxPl{$n}) { $maxPl++; }
        }
      } else {
         unless ($c->param($e)=~/^null$/i || $ph{$e} ne "")
         {
            $e_neu=$e;
            if ($e=~/^[dn]/) {
```

```
                $e_neu=~s/^(.)//;
                &syntaxCheck($1,$c->param($e));    # Syntax
                    ueberpruefen
             }
             $ph{$e_neu}=&cgi2ascii($c->param($e));
          }
       }
    }
 }
}
```

Auswerten der Konfigurationsfiles:

```
sub checkConfig {
   open(FD,$_[0]) || printError("Datei nicht gefunden",
       "Datei $_[0] konnte nicht ge&oumlffnet werden.");

   $status=0;
   $scriptLevel=1;           # Genaue Sicherheitspruefung

   while(<FD>) {
      chop;

      if ($status==0 && /^\s*(\w+)\s*\{/) {   # Neuer Eintrag
         $typ=$1;
         $status=1;

         s/^\s*\w+\s*\{\s*//;
         $zeile=$_;
      } elsif ($status==1) {     # Textzeile
         $zeile=$zeile." ".$_;
      }

      if ($status==1 && /\}/) {   # Ende eines Eintrages
         $zeile=~s/\s*\}.*$//;

         # Eintrag auswerten

         if ($typ=~/^S$/) { $mainSQL=$zeile;}       # SQL-
            Statement (P)
         elsif ($typ=~/VF/) { $formFile=$zeile;}  # Vorlagen-
            Datei
         elsif ($typ=~/HL/) { $formHead=$zeile;}  #
            Seitenkopfzeile
         elsif ($typ=~/TL/) { $formTitle=$zeile;} #
            Seitentitel
         elsif ($typ=~/LE/) { $element=$zeile;}   #
            Listenelementvorlage (P)
         elsif ($typ=~/^H$/) { $preText=$zeile; }   #
            Seitenkopftext
         elsif ($typ=~/^T$/) { $postText=$zeile; }  #
            Seitenfusstext
```

```
            elsif ($typ=~/SH/) { $headSQL=$zeile; }  # Kopf-SQL-
               Statement (P)
            elsif ($typ=~/P/) {                      # Parameter
               lesen
               foreach $e (split(/\s+/,$zeile)) {
                  ($n,$v)=split(/\=/,$e);
                  unless ($v=~/^null$/i || $ph{$n} ne "") {
                     $ph{$n}=&cgi2ascii($v);
                  }
               }
            }
            elsif ($typ=~/E0/) { $errorNull=$zeile;} # Keine
               Saetze gefunden
            elsif ($typ=~/EM/) { $errorMain=$zeile;} # Fehler im
               Haupt-SQL-smt
            elsif ($typ=~/EH/) { $errorHead=$zeile;} # Fehler im
               Head-SQL-Smt
            elsif ($typ=~/^L$/) { $scriptLevel=$zeile;} #
               Sicherheitsstatus

            $status=0;
         }
      }
   }
```

4. Screenshots / Appearance-Vergleiche

Im folgenden wird die Auswahlmaske „Prüfungen selektieren“, wie sie unter Windows95, Unix und MacOS angezeigt wird, dargestellt.

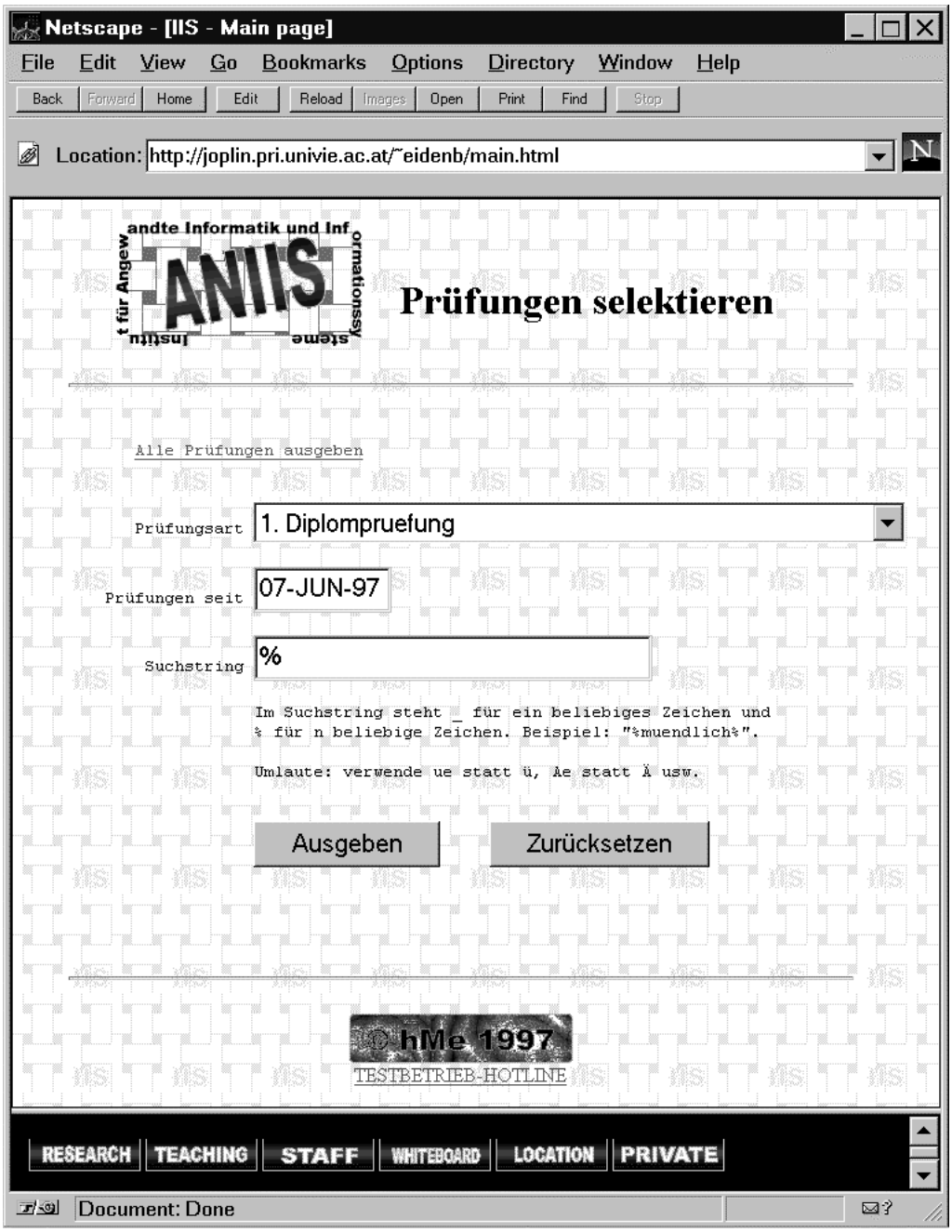

Abb. 34: Suchmaske unter Windows 95

Im Unterschied zur Windows-Maske ist die Unix-Maske breiter; das liegt wohl daran, daß Unix eine andere Courier-Schriftart verwendet. Außerdem werden natürlich die Auswahlobjekte anders dargestellt:

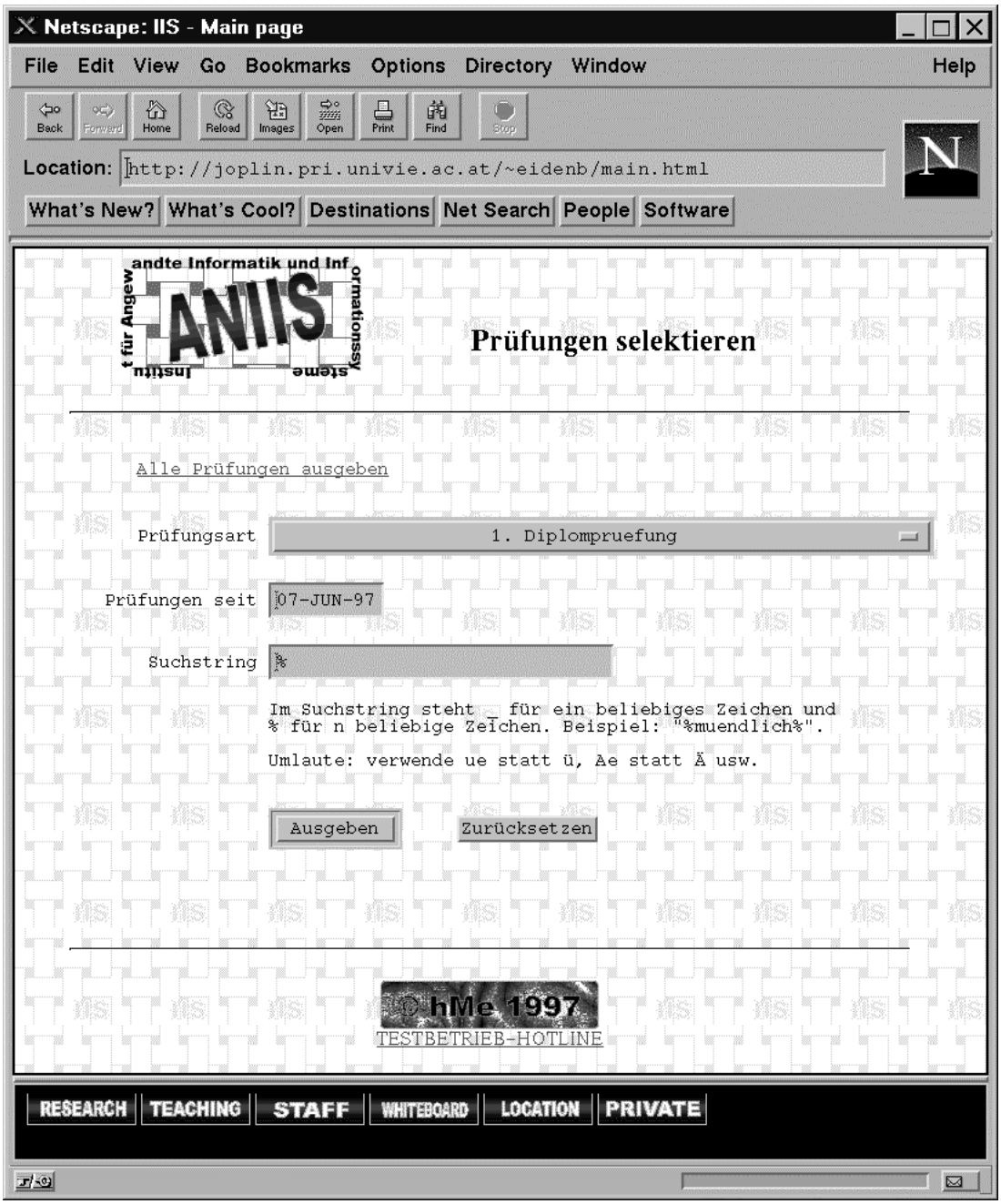

Abb. 35: Suchmaske unter Solaris (auf Windows-Client dargestellt)

Am Macintosh ist der Hintergrund heller und die Buttonleiste wird vergleichsweise groß dargestellt:

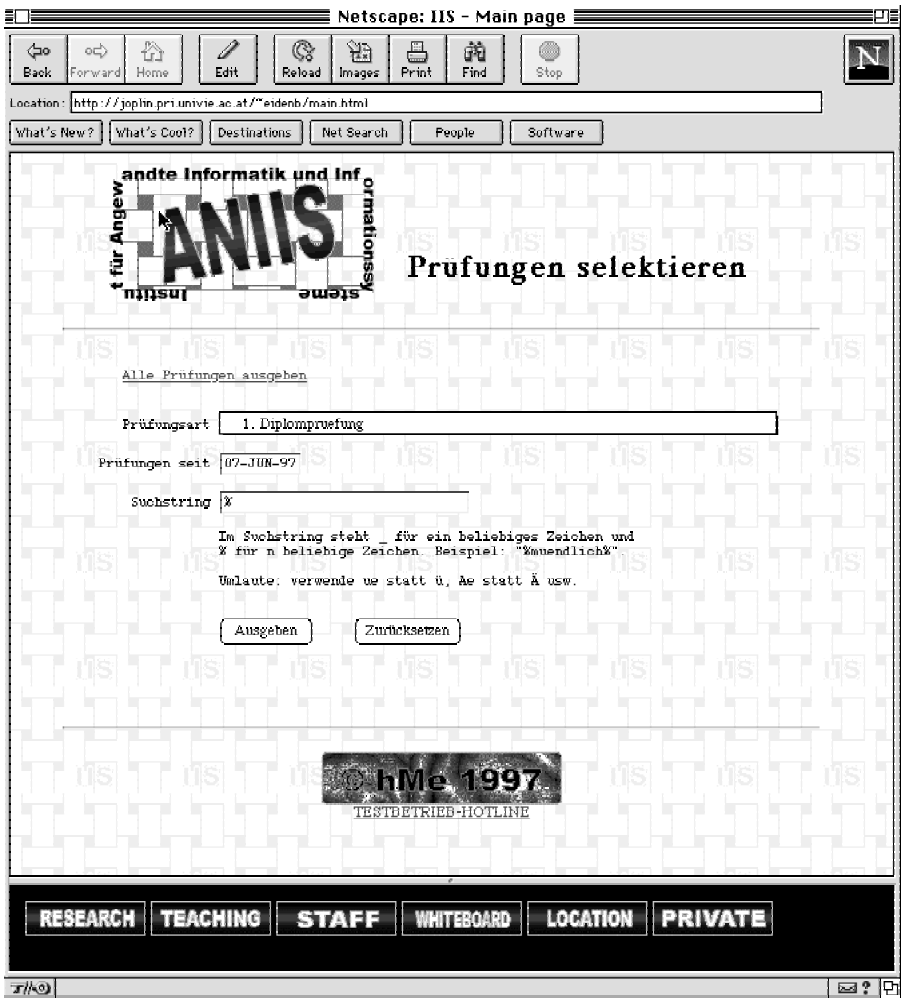

Abb. 36: Suchmaske am Mac

Als zweiter Vergleich wird eine Lehrveranstaltungskarte angezeigt. Da hier keine Widgets vorkommen, sehen die Screenshots auf allen drei Systemen relativ gleich aus:

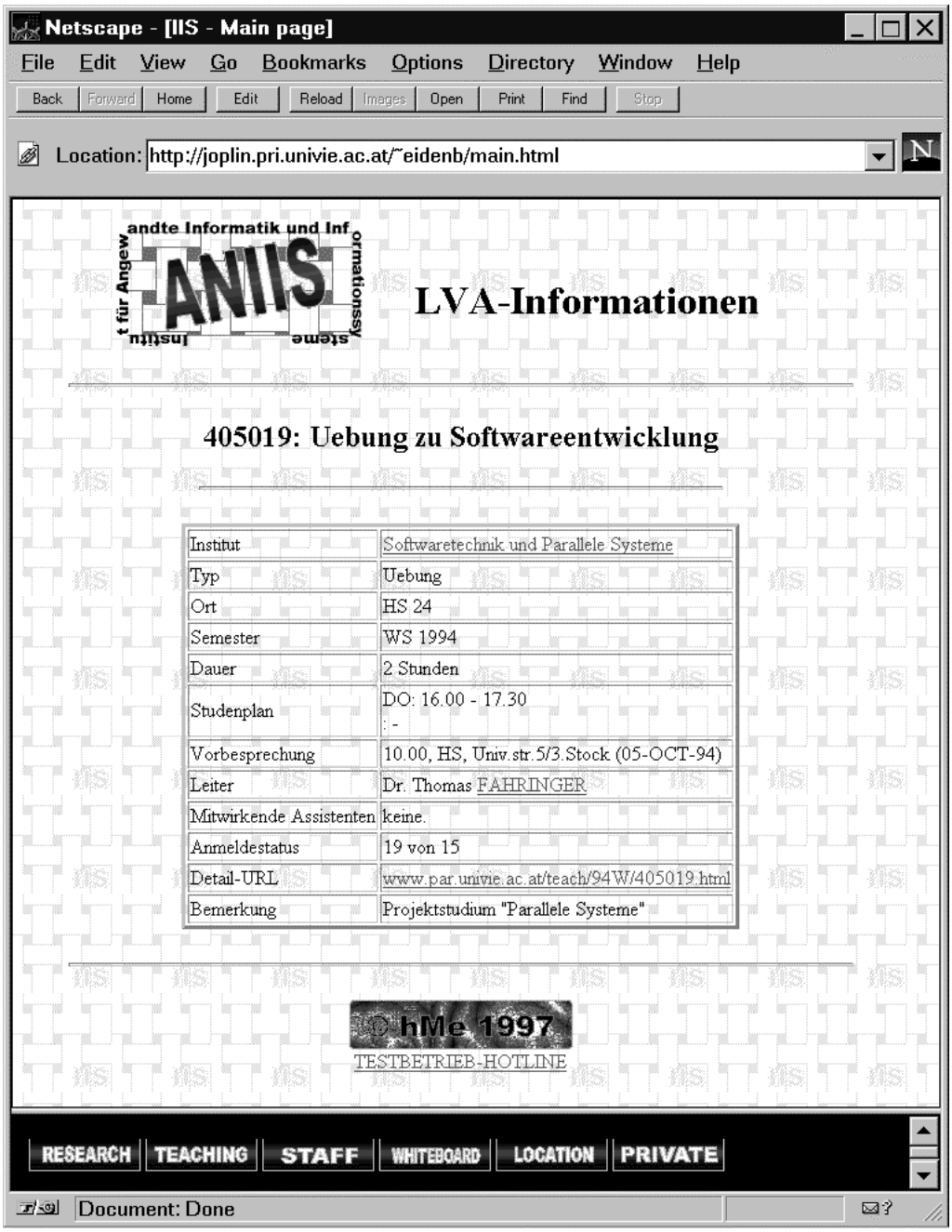

Abb. 37: LVA-Karte unter Windows 95

Diesmal wirkt die Unix-Darstellung aufgrund der Schriftart kompakter. Interessanterweise wird die Überschrift („<h1>LVA-Informationen</h1>“) kleiner als die Kartenüberschrift („<h2>405019 ...</h2>“) dargestellt:

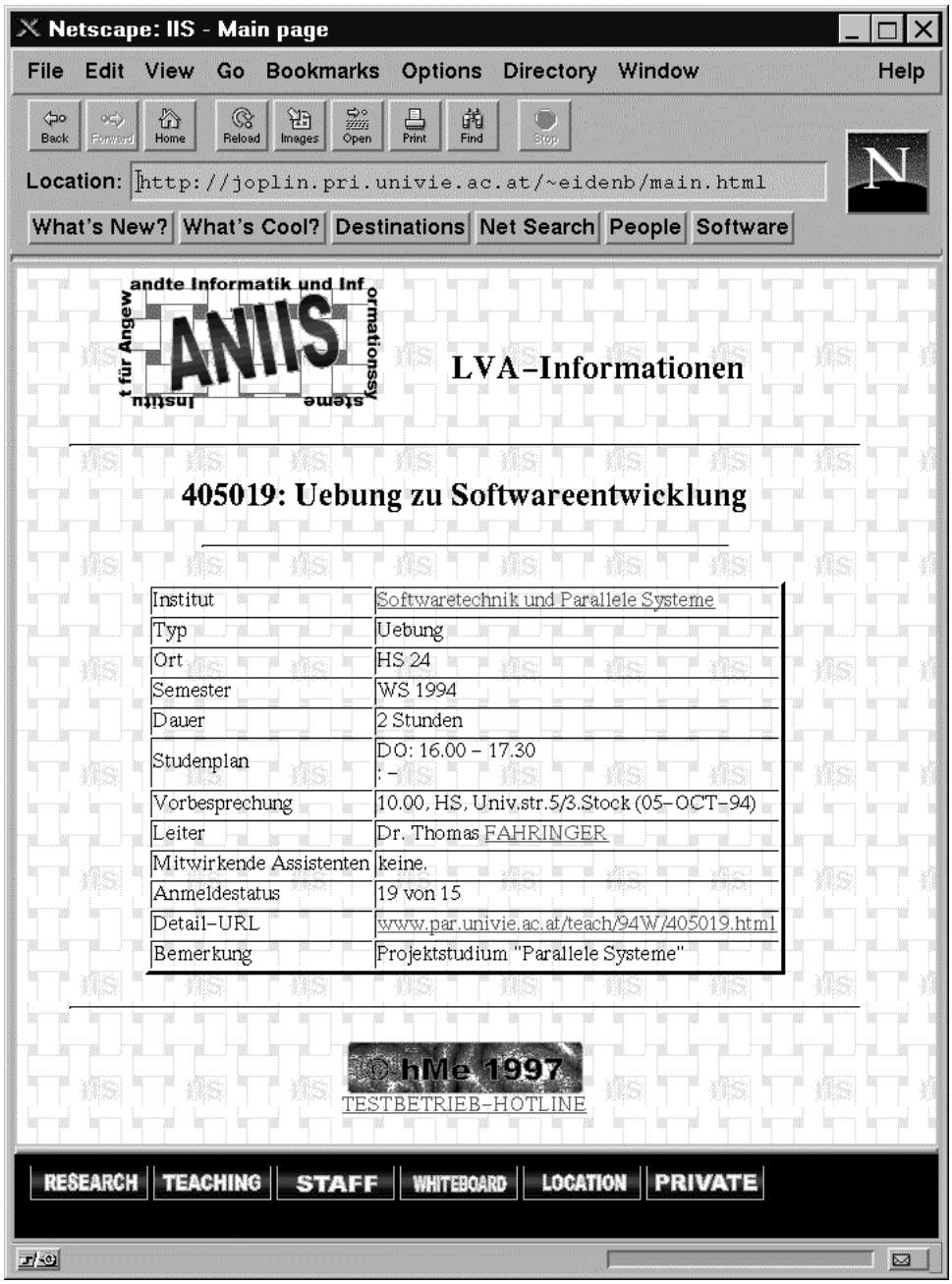

Abb. 38: LVA-Karte unter Solaris

Am Mac werden Schriften wieder kleiner und Grafiken größer ausgegeben:

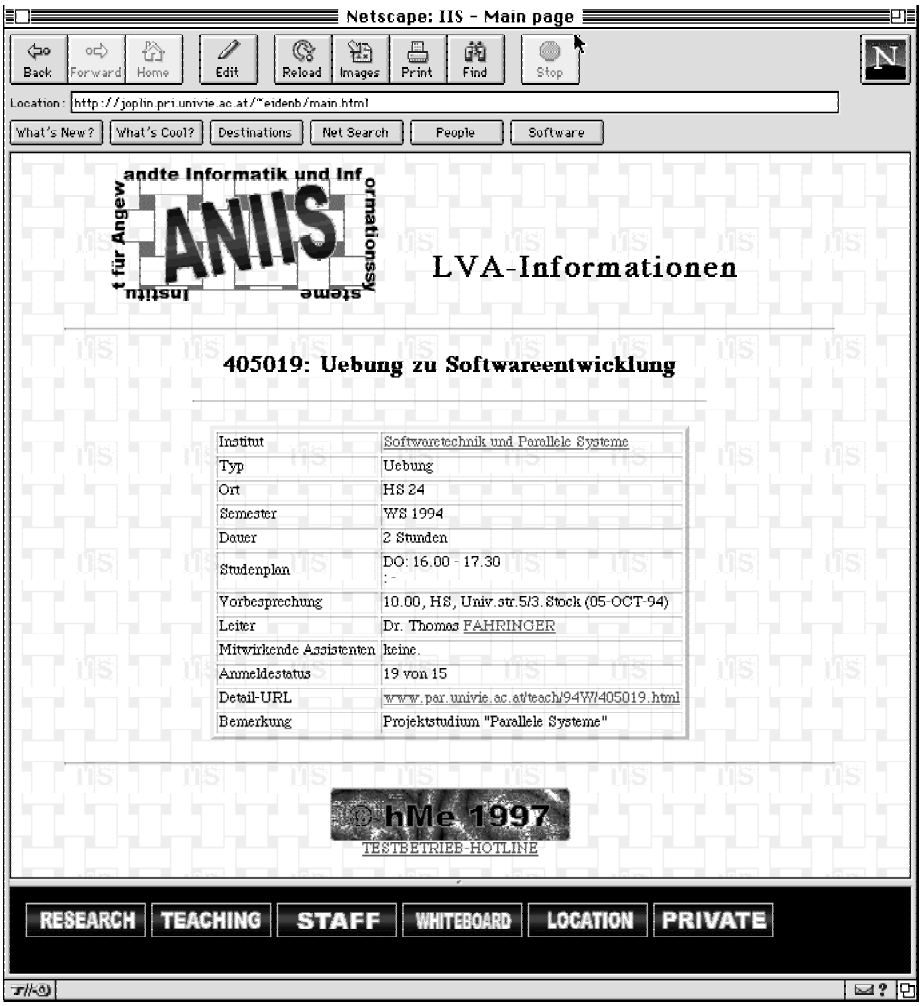

Abb. 39: LVA-Karte am Mac

5. Dokumentierte Konfigurationsdateien- und Vorlagenliste

Vorlage	Beschreibung
abmLVA.cfg	Lehrveranstaltungen abmelden
abmPR.cfg	Prüfungen abmelden
addPub.cfg	Publikation hinzufügen
addPubautor.cfg	Publikationsautoren hinzufügen
addWHB.cfg	Whiteboardeintragsmaske ausgeben
addWHB.html	Whiteboardeintrag hinzufügen - Maske
addWHBEntry.cfg	Whiteboardeintrag hinzufügen
addWHK.cfg	Whiteboardkategorienmaske ausgeben
addWHK.html	Whiteboardkategorie hinzufügen - Maske
addWHKEntry.cfg	Whiteboardkategorie hinzufügen
anmLVA.cfg	Lehveranstaltungsstatus ausgeben
anmPR.cfg	Prüfungsstatus ausgeben
chgPub.cfg	Publikationseintrag ändern
chgStudDaten.cfg	Studentenstammdaten ändern
chgStudPwd.cfg	Studentenpasswort ändern
chgWHB.cfg	Whiteboardeintrag ändern - Maske ausgeben

chgWHB.html	Whiteboardeintrag ändern - Maske
chgWHBEntry.cfg	Whiteboardeintrag ändern
chgWHK.cfg	Whiteboardkategorie ändern - Maske ausgeben
chgWHK.html	Whiteboardkategorie ändern - Maske
delCaps.cfg	Alle aktuellen Capabilities löschen
delPub.cfg	Publikation löschen
delWHB.cfg	Whiteboardeintrag löschen
delWHBSystem.cfg	Abgelaufene Whiteboardeinträge löschen
delWHBZensur.cfg	Whiteboard-Zensur für IIS-Manager
delWHK.cfg	Whiteboardkategorie löschen
getKategorien.cfg	Alle Prüfungskategorien ausgeben
getLktNames.cfg	Alle Lektorennamen ausgeben
homepage.cfg	IIS-Hauptseite ausgeben
lektLogin.cfg	Lektoren-Login-Maske ausgeben
lektLogin.html	Lektoren-Login-Maske
lektMenu.cfg	Lektoren-Menü ausgeben
lektMenu.html	Lektoren-Menü
listAbmLVA.cfg	Liste aller angemeldeten LVAs ausgeben
listAbmPR.cfg	Liste aller angemeldeten Prüfungen ausgeben
listAllLVA.cfg	Alle Lehrveranstaltungen ausgeben
listAllPR.cfg	Alle Prüfungen ausgeben
listAllPub.cfg	Alle Publikationen ausgeben
listAnmLVA.cfg	Alle nicht-angemeldeten LVAs ausgeben
listAnmPR.cfg	Alle nicht-angemeldeten Prüfungen ausgeben
listLVA.cfg	Lehrveranstaltungen selektiv ausgeben
listLVALeiter.cfg	Lehrveranstaltungsleiter ausgeben
listLVAMit.cfg	LVA-Mitwirkende als Liste ausgeben
listPR.cfg	Prüfungen selektiv ausgeben
listPRAufs.cfg	Prüfungs-Aufsichtspersonen ausgeben
listPRLeiter.cfg	Prüfungsleiter ausgeben
listPRMit.cfg	Prüfungsmitwirkende ausgeben
listPRTypen.cfg	Prüfungstypen ausgeben
listPub.cfg	Publikationen selektiv ausgeben
listPubAut.cfg	Publikationsautoren ausgeben
listPubChg.cfg	Publikationen zum Ändern auflisten
listPubDel.cfg	Publikationen zum Löschen auflisten
listSelAnmPR.cfg	Nicht-angemeldete Prüfungen selektiv ausgeben
listSelLVA.cfg	Lehrveranstaltungen selektiv ausgeben
listSelPR.cfg	Prüfungen selektiv ausgeben
listStudienjahre.cfg	Alle Studienjahre ausgeben
listWHBAll.cfg	Alle Whiteboardeinträge ausgeben
listWHBChg.cfg	Whiteboardeinträge zum Ändern ausgeben
listWHBDel.cfg	Whiteboardeinträge zum Löschen ausgeben
listWHBEntry.cfg	Whiteboardeinträge selektiv ausgeben
listWHBZensur.cfg	Whiteboard für Zensur ausgeben
listWHKChg.cfg	Whiteboardkategorien für Änderung ausgeben
listWHKDel.cfg	Whiteboardkategorien für Löschen ausgeben
logout.cfg	Logout
research.cfg	Research-Seite

selAnmPR.cfg	Nicht-angemeldete Prüfungen auswählen - Maske anzeigen
selAnmPR.html	Nicht-angemeldete Prüfungen auswählen - Maske
selLVA.cfg	LVA auswählen - Maske anzeigen
selLVA.html	LVA auswählen - Maske
selPR.cfg	Prüfungen auswählen - Maske anzeigen
selPR.html	Prüfungen auswählen - Maske
selPub.cfg	Publikationen auswählen - Maske anzeigen
selPub.html	Publikationen auswählen - Maske
selWHB.cfg	Whiteboardeinträge auswählen - Maske anzeigen
selWHB.html	Whiteboardeinträge auswählen
showLVA.cfg	LVA-Änderungsmaske anzeigen
showLVA.html	LVA-Änderungsmaske
showPR.cfg	Prüfungen-Änderungsmaske anzeigen
showPR.html	Prüfungen-Änderungsmaske
showPubAdd.cfg	Publikationen hinzufügen - Maske anzeigen
showPubAdd.html	Publikationen hinzufügen - Maske
showPubChg.cfg	Publikationen ändern - Maske anzeigen
showPubChg.html	Publikationen ändern - Maske
showStaffList.cfg	Mitarbeiterliste ausgeben
showStaffMember.cfg	Mitarbeiterkarte anzeigen
showStaffMember.html	Mitarbeiterkarte
showStud.cfg	Studentenstammdaten anzeigen
showStud.html	Studentenstammdaten - Maske
showStudPwd.cfg	Passwort-Änderungsmaske anzeigen
showStudPwd.html	Passwort-Änderungsmaske
showWHBChg.cfg	Whiteboardeintrag ändern - Maske ausgeben
showWHBChg.html	Whiteboardeintrag ändern - Maske
showWHKChg.cfg	Whiteboardkategorien ändern
studLogin.cfg	Studenten-Login-Maske ausgeben
studLogin.html	Studenten-Login-Maske
studMenu.cfg	Studenten-Menü ausgeben
studMenu.html	Studenten-Menü
teaching.cfg	Teaching-Seite ausgeben

Tab. 24: Dokumentierte Vorlagenliste

6. Dokumentierter Sourcetree

Verzeichnis	Bedeutung
/	Homeverzeichnis des Systemes
/sql	Grundverzeichnis der Datenbankdateien
/sql/alt	SQL-Dateien für das aktuelle IIS
/sql/neu	SQL-Dateien für das erweiterte IIS
/www	WWW-Grundverzeichnis
/www/cgi-bin	Interpreter-Verzeichnis
/www/develop	Verschiedene Weiterentwicklungen
/www/generiert	Generierte Dateien

/www/graphics	Graphikdateien
/www/graphics/eidi-set	Derzeit verwendetes Graphikset
/www/graphics/inst-set	Im WWW-Konzept vorgeschlagens Set
/www/graphics/misch-set	Alternatives Set
/www/standard	Standard-Konfigurationsdateien aller Art
/www/vorlagen	Konfigurationsdateien und Vorlagen

Tab. 25: Dokumentierter Sourcetree