Universität Hamburg Fachbereich Informatik Arbeitsbereich Softwaretechnik

Studienarbeit

EINE FALLSTUDIE ZUR OBJEKTORIENTIERTEN DOMÄNENANALYSE

Vorgelegt von:

Christian Fürll
Holger Koschek
Thomas Pfohe

Betreuerinnen:

Prof. Dr. Christiane Floyd
Anita Krabbel

Inhaltsverzeichnis

Vorwort

Diese Studienarbeit stellt den ersten Versuch an der Universität Hamburg dar, eine Domänenanalyse durchzuführen. Motiviert hat uns dazu in erster Linie Frau Prof. Dr. Christiane Floyd, der wir dafür recht herzlich danken. Besonders danken möchten wir Anita Krabbel, die uns bei Umsetzung und Durchführung dieser Studienarbeit immer hilfreich zur Seite stand.
Nicht vergessen sollten wir die übrigen TeilnehmerInnen des Projektseminars „Ein Objektorientiertes Entwurfsschema für STEPS“ im Sommersemester 1994, aus dem diese Studienarbeit entstand.

Hamburg, im April 1996

1 Einführung

Diese Studienarbeit beschäftigt sich mit einer Fallstudie zum Komplex der objektorientierten Domänenanalyse. Eine Einführung in dieses Gebiet geben wir im nächsten Abschnitt. Hier soll es zunächst darum gehen, in welchem Umfeld unsere Arbeit entstand und was für eine Zielsetzung wir haben.

1.1 Das Umfeld

Im Sommersemester 1994 wurde von Frau Prof. Dr. Christiane Floyd, Professorin für Softwaretechnik an der Universität Hamburg, ein Projektseminar mit dem Titel „Ein Objektorientiertes Entwurfsschema für STEPS“ angeboten. Es ging dabei grundlegend um die Fragestellung, ob und wie STEPS (**S**oftware**t**echnik für **E**volutionäre, **P**artizipative **S**ystemgestaltung)[1] und die dazu angebotene Vorlesung „Einführung in die Softwaretechnik“ in ein objektorientiertes Schema überführt bzw. erweitert werden können. Bis dahin wurde STEPS noch mit einem rein objektbasierten, modularen Aufbau vertreten. Das Hauptziel war es, ein Verständnis für die (Weiter-) Entwicklung methodischer Ansätze von STEPS in Richtung *Objektorientierung* zu erlangen. Außerdem sollte bei den TeilnehmerInnen[2] ein Grundwissen über objektorientierte Ansätze und die Sprache Eiffel[3] aufgebaut werden. Die Aufgabenstellung eines Programmierpraktikums an der TU Berlin im Sommersemester 1991[4] und eine darauf aufbauende programmiertechnische Studienarbeit wurden als Grundlage zum Erarbeiten dieses Themenkomplexes ausgewählt. In dem erwähnten Praktikum ging es um den Aufbau eines Systems zur computergestützten Erstellung eines Vorlesungsverzeichnisses (VV-Erstellung).
Das Projektseminar fand teils im Plenum und teils in Kleingruppen statt. Diese Teams sind von den Mitveranstaltern geleitet worden. Es entstanden Gruppen mit folgenden Themenschwerpunkten:

- Anforderungsermittlung, Domänenmodellierung
  Leiter: Anita Krabbel, Wolfgang Strunk
- Systemgestaltung und Softwareentwurf
  Leiterin: Christiane Floyd
- Softwareentwurf und Programmierung
  Leiter: Peter von Savigny

Diese Kleingruppen erschlossen sich selbständig die zu ihrem Thema notwendige Literatur und untersuchten die vorgegebene Fallstudie anhand der technischen Dokumentation und der am Rechner verfügbaren Implementation im Hinblick auf veränderte Anforderungen, bezogen auf die oben genannten Zielsetzungen des Seminars.
Im Plenum wurden Grundlagen vermittelt, Referate zu den einzelnen Schwerpunkten gehalten und die Untersuchungsergebnisse der Fallstudie zusammengetragen.

1.2 Zur Stellung der Studienarbeitsverfasser im Projektseminar

Wir, die Autoren dieser Studienarbeit, gehörten mit drei weiteren Kommilitonen zur Gruppe Anforderungsermittlung und Domänenmodellierung. Dies bedeutet, daß insgesamt 8 Teilnehmer dieser Kleingruppe an dem Komplex der Domänenmodellierung arbeiteten. Auf die genaue

[1] vgl. Floyd, Christiane: Arbeitsunterlagen zur Lehrveranstaltung „Einführung in die Softwaretechnik“, Universität Hamburg, Ausgaben 1991 bis 1994

[2] Wir sind uns bewußt, daß sich die Leserschaft dieser Studienarbeit sowohl aus weiblichen als auch aus männlichen Personen zusammensetzt. Wegen der leichteren Lesbarkeit verzichten wir aber ab hier auf diese umständlichere Schreibweise.

[3] vgl. Meyer, Bertrand: Objektorientierte Softwareentwicklung, Prentice Hall 1990

[4] siehe Anhang C

Struktur dieser Kleingruppe wird später noch detailliert eingegangen, doch hier soll vorweg schon folgendes beschrieben werden: Anita Krabbel und Wolfgang Strunk übernahmen Leitung und Planung der Kleingruppe; Stephan Fröhlich, Oliver Satz und Marko Fricke arbeiteten sich in den Vorgang des Erstellens eines *Vorlesungsverzeichnisses* (VV) in Berlin ein, und wir betrachteten als weiteren Anwendungsfall die VV-Erstellung am Fachbereich Informatik der Universität Hamburg.

1.3 Zur Zielsetzung dieser Studienarbeit

Diese Studienarbeit hat als Ziel, dem Leser einen Einblick in unseren Vorgehensprozeß bei der VV-Erstellung zu geben. Dies geschieht, indem wir den Ablauf der Szenario-Erstellung und der daraus erarbeiteten Anforderungsermittlung schildern. Darauf aufbauend führen wir eine Synthese und anschließend einen kritischen Vergleich der verschiedenen Vorgänge bei der VV-Erstellung in Hamburg und Berlin durch und versuchen, eine objektorientierte „Domäne“[1] zu bilden. Daher entspricht der gliederungstechnische Aufbau dieser Studienarbeit auch weitestgehend dem zeitlichen Projektverlauf. Gleichzeitig sollen aber auch die verschiedenen Zwischenergebnisse immer wieder bewertet und mit den Zielsetzungen der Domänenanalyse, wie sie in der Literatur beschrieben sind, verglichen und beurteilt werden.
Zusätzlich werden wir in späteren Kapiteln auch auf die Beziehung zwischen Objektorientierung und STEPS eingehen. Wir werden versuchen, herauszuarbeiten, wo sich unserer Meinung nach Konzepte und Ideen ergänzen, oder wo sie sich unterscheiden. Diese Arbeit ist daher auch immer unter dem Blickwinkel zu sehen, wie STEPS momentan gelehrt wird.

1.4 STEPS - Eine kurze Einführung

Das Ziel des angesprochenen Projektseminars war es, ein objektorientiertes Entwurfsschema für STEPS zu entwickeln. An dieser Stelle soll deshalb STEPS, wie es bisher verstanden wurde, von uns dargestellt werden, um dem Leser einen kurzen Überblick über diese Methode zu geben. Nachfolgend werden wir uns nämlich noch des öfteren auf diese Softwareentwicklungsmethode berufen, was in der Thematik unserer Studienarbeit begründet liegt. Diese Erläuterung erhebt erwartungsgemäß keinen Anspruch auf Vollständigkeit; dazu sind die Arbeitsunterlagen zur Vorlesung „Einführung in die Softwaretechnik“[2] sehr viel besser geeignet. Die in diesem Abschnitt ausgeführten Punkte sind daher auch nur als kurze Zusammenfassung einiger wesentlicher Aspekte von STEPS zu sehen.
STEPS (**S**oftware**t**echnik für **E**volutionäre **P**artizipative **S**ystemgestaltung) ist als eine Methode bzw. Sichtweise zu verstehen, „bei der die Softwareentwicklung als Kommunikations- und Lernprozeß gesehen wird, in dem verschiedene Personengruppen (Hersteller, Systemanalytiker und Benutzer) mit unterschiedlichem Vorwissen und unterschiedlichen Interessen zusammenarbeiten und in dem eine Folge von aufeinander aufbauenden Versionen des Softwaresystems entwickelt und benutzt werden (Evolution)“.[3] Ein weiterer wichtiger Punkt dieses Ansatzes ist, daß das zu schaffende Softwaresystem den Benutzer „sinnvoll“ und „angemessen“[4] unterstützen soll. Um diese Ziele zu erreichen, wird davon ausgegangen, daß die Partizipation des späteren Benutzers während der Entwicklung wesentlich ist. Das Anwendungsgebiet, welches von STEPS besonders betrachtet wird, sind Systeme, die in menschliches Arbeitshandeln integriert sind. Programme werden also immer im direkten Zusammenspiel mit dem Benutzer betrachtet und nicht als ganz und gar eigenständige Einheiten. STEPS als Methode zur Softwareentwicklung wird in dem Zusammenhang der angesprochenen Vorlesung „Einführung in die

[1] siehe Kapitel 2.2.2
[2] vgl. Floyd (a.a.O.)
[3] vgl. Floyd (a.a.O.), S.18
[4] vgl. Floyd (a.a.O.), S.18

Softwaretechnik“ natürlich mit speziellen Techniken, Werkzeugen und Organisationsformen[1] unterlegt, die diese Methode unterstützen. Aus zweierlei Gründen sei darauf hier nicht weiter eingegangen:

1. Diese speziellen praktischen Umsetzungen der Methode STEPS in den Bereichen Anforderungsermittlung, Systemspezifikation, Benutzungsschnittstellen und Softwareentwurf werden in dieser Studienarbeit teilweise dann kurz vorgestellt, wenn wir sie benötigen. Ein Beispiel dafür sind sicherlich die Szenarios und Aufgabennetze bei der Anforderungsermittlung (Siehe Kapitel 3.1).
2. Es werden von uns zum Teil andere Techniken benutzt, die in den Arbeitsunterlagen zu „Einführung in die Softwaretechnik“ nicht beschrieben werden. Dies ist dadurch begründet, daß bisher STEPS mit einem modularen, objektbasierten Aufbau vertreten wurde, wir aber hier zur Objektorientierung übergehen werden, was die Anwendung anderer Techniken voraussetzt. Dies ist kein Widerspruch, denn eine Umsetzung mit anderen Techniken und Werkzeugen kann durchgeführt werden, „ohne das Gesamtanliegen der Methode zu verletzen“.[2]

Wichtig herauszustellen ist bei diesem Ansatz also vor allem, daß der **Prozeß** der Softwareentwicklung unterstützt werden soll. Bei anderen Ansätzen steht hingegen eine produktorientierte Sichtweise im Vordergrund, die von festen Anforderungen ausgeht. Bei diesen Ansätzen handelt es sich dann um sogenannte Phasenmodelle,[3] die ein Softwaresystem Top-Down mit verschiedenen Zwischenstufen entwickeln wollen. Die Hauptunterschiede zu den Top-Down-Herangehensweisen möchten wir hier kurz darlegen:[4]

- Es wird keine vordefinierte Herangehensweise propagiert, sondern in konkreten Situationen werden relevante Strategien gewählt.
- Es wird keine lineare, sondern eine zyklische Herangehensweise angemahnt, bei welcher der Prozeß der Softwareentwicklung die Herstellung und den Einsatz von Software miteinander verknüpft.
- Es wird von der reinen Top-Down-Strukturierung zu einer schrittweisen Vorgehensweise übergegangen, die neben Verfeinerung und Formalisierung auch bewußt Revisionen aufgrund weitergehender Erkenntnisse und Einsichten zuläßt.
- Es wird die Kooperation und Kommunikation zwischen allen Beteiligten betont.
- Es wird von der einmaligen Erstellung von Dokumenten zu einer versionsorientierten Sicht gewechselt.

Diese Übersicht sollte reichen, um den weiteren Ausführungen dieser Studienarbeit, wenn sie sich auf STEPS beziehen, folgen zu können.

Eine Verschränkung soll hier noch angemerkt werden: Wir bewegen uns mit unserer Studienarbeit nicht nur im (objektorientierten) STEPS-Kontext, sondern vor allen Dingen in der Problematik, uns mit der objektorientierten Domänenanalyse auseinanderzusetzen. Dadurch werden einige von STEPS geforderte Bereiche, wie z.B. die reale Softwareumsetzung, von uns nicht geleistet. Dafür werden aber gerade die Anforderungsermittlung und die Modellbildung einen Großteil dieser Arbeit ausmachen.

[1] vgl. Floyd (a.a.O.), S.15f
[2] vgl. Floyd (a.a.O.), S.21
[3] vgl. Kimm, Koch, Simonsmeier, Tontsch: Enführung in Software Engineering, Walter de Gruyter, Berlin 1979; zitiert bei: Floyd (a.a.O.), S.31f
[4] vgl. Floyd (a.a.O.), S.40f

2 Überblick über die Domänenanalyse

Dieser Abschnitt gibt einen kurzen Überblick über die Begriffe, Voraussetzungen und Methoden der *Domänenanalyse* (DA), um die Grundlagen der Studienarbeit zu beschreiben. Ein Großteil der hier angesprochenen Themen schälte sich im Projektseminar „Ein objektorientiertes Entwurfsschema für STEPS“ im Sommersemester 1994 heraus.
Die Literaturarbeit in diesem Kapitel beruht hauptsächlich auf zwei Veröffentlichungen von Prieto-Díaz&Arango[1] und Berard.[2] Da selbst grundlegende Begriffe der DA nicht allgemein anerkannt definiert sind, werden wir – wo es nötig ist – die unterschiedlichen Auffassungen von Prieto-Díaz&Arango und Berard anführen und unsere Interpretation daran begründen.

2.1 Motivation der DA

Für die Entwicklung von Software ist die Analyse des Anwendungsbereiches der zukünftigen Software unumgänglich. Der Prozeß der Domänenanalyse wurde entwickelt, um die Erkennung und Abbildung der Informationen über Elemente[3] des Anwendungsbereiches und die Beziehungen dieser Elemente in speicherbare Form zu verbessern. Bei der Domänenanalyse untersucht der *Domänenanalytiker* (siehe Kapitel 2.7) verschiedene Beispiele aus einem Anwendungsbereich, um so Elemente und Operationen aus dem Anwendungsbereich zu gewinnen, welche den Beispielen gemeinsam und somit möglichst allgemein verwendbar sind. So werden wir in unserem Anwendungsbereich „Erstellung eines Vorlesungsverzeichnisses“ sowohl die Realisierung in Berlin als auch in Hamburg untersuchen, um Detail- und Strukturwissen[4] über in beiden Beispielen verwendbare Elemente zu finden.

Mit der DA ist somit der Begriff der *Wiederverwendbarkeit* (reusability/reuse) stark verknüpft. Die DA soll u.a. die Wiederverwendbarkeit von Software systematisch verbessern helfen. Eine möglichst hohe Wiederverwendbarkeit wird angestrebt, da man sich durch sie folgende Verbesserungen erhofft:[5]

- geringere Entwicklungskosten
- geringere Produktkosten
- höhere Produktivität
- Risikominderung
- verkürzte Entwicklungszeit
- größere Zuverlässigkeit
- größere Fehlerfreiheit
- größere Effizienz der Software
- verbessertes rapid prototyping

---

[1] vgl. Rubén Prieto-Díaz, Guillermo Arango: Domain Analysis Concepts and Research Directions, in: Rubén Prieto-Díaz, Guillermo Arango: Domain Analysis and Software Systems Modeling, IEEE 1991

[2] vgl. Edward V. Berard: Object-Oriented Domain Analysis, in: Edward V. Berard: Essays on Object-Oriented Software Engineering, 1993

[3] Diesen Begriff verwenden wir im folgenden, um nicht das schon sehr mit Bedeutung befrachtete Wort „Objekt“ benutzen zu müssen. Der Begriff „Element“ soll durch erkennbare Eigenschaften eindeutig identifizier- und unterscheidbare Einheiten bezeichnen.

[4] Aus der Systemtheorie hat sich für „Informationen über die Elemente eines Systems“ der Begriff *Detailwissen*, für „Informationen über die Beziehungen von Elementen in einem System“ der Begriff *Strukturwissen* gebildet (vgl. Bernd Page: Diskrete Simulation, Berlin u.a. 1991, S. 1).

[5] vgl. Berard (a.a.O.), S. 184

Durch die Verschmelzung mit Konzepten aus der Objektorientierung[1] (OO) wurde aus der DA die Objektorientierte Domänenanalyse (OODA). So ist die OODA vorrangig auf das Erkennen der Objekte des Anwendungsbereiches ausgerichtet.

Die DA ist im Vergleich mit anderen Software-Entwicklungsmethoden noch nicht sehr weit entwickelt. Inwieweit STEPS zusammen mit OO und DA und somit OODA harmoniert, wird ebenfalls in dieser Fallstudie untersucht.

2.2 Grundlegende Begriffe der DA

2.2.1 Problemdomäne

Für uns wird eine *Problemdomäne* durch eine Menge von Merkmalen charakterisiert, welche eine Problemklasse umfassend und genau beschreibt.
Eine Problemdomäne ist losgelöst von einer Ausrichtung (z.B. Softwareerstellung); sie ist lediglich ein Mittel, um eine Problemklasse wertneutral darstellen zu können.

2.2.2 Domäne

Im Allgemeinen versteht man unter *Domäne* den Begriff *Anwendungsdomäne*[2], also z.B. Textverarbeitungsprogramme, GUIs oder in unserem Fall Programme zur Erstellung von Vorlesungsverzeichnissen. Aber auch nicht softwarebezogene Gebiete fallen unter diesen Begriff, so ist z.B. „Erstellung von Vorlesungsverzeichnissen“ auch dann eine Domäne, wenn dazu noch keine Programme verwendet werden.

Genauer ist eine Domäne

1. eine Sammlung von aktuellen und zukünftigen (Software-) Anwendungen, die eine Menge von gemeinsamen Merkmalen teilen

oder

2. eine Menge von Merkmalen, welche eine Problemklasse, für die Softwarelösungen gesucht werden, umfassend und genau beschreibt.[3]

Im Gegensatz zur Problemdomäne ist eine Domäne auf die Softwarekonstruktion ausgerichtet. Domänen sind die Teilmengen der Problemdomänen, für die mit Hilfe der DA Software entwickelt wird oder werden soll.

Da man für die DA eine Gruppe von Domänenanalytikern benötigt, werden verschiedene solcher Gruppen verschiedene Sichten auf eine Domäne haben: Es besteht eine Gruppenabhängigkeit der Domäne.

Die Begriffe Domäne und Problemdomäne werden häufig synonym verwendet, zumal es keine allgemein akzeptierte Definition dieser Begriffe gibt. So definieren Prieto-Díaz&Arango den Begriff Problemdomäne wie folgt:

[1] Eine Definition des Begriffes „Objektorientierung“ finden Sie in Kapitel 5.1.
[2] vgl. Berard (a.a.O.), S. 184f
[3] vgl. Berard (a.a.O.), S. 185

„For the sake of simplicity, we have used the expression problem domain as a synonym for a class of problems. However, the term does not refer to any arbitrary class of problems and related information. In the context of our discussion, a body of information is considered a problem domain if

- deep or comprehensive relationships among the items of information are known or are suspected with respect to some class of problems,
- there is a community that has a stake in solving the problems,
- the community seeks software-intensive solutions to these problems, and
- the community has access to knowledge that can be applied to solving the problems.“[1]

Nach dieser Definition ist die Problemdomäne eine Spezialisierung der Domäne. Wir verwenden im folgenden aber unsere erste Definition.

2.2.3 Domänenmodell

Das *Domänenmodell* ist das Ergebnis der DA. Allgemein gesprochen ist es ein graphisches, textuelles oder ausführbares Modell der Domäne. Die Auffassungen über die genaue Ausprägung des Begriffs Domänenmodell unterscheiden sich sowohl in der Definition als auch in der Art und Weise, wie es erstellt werden soll.
Berard definiert den Begriff Domänenmodell nicht. Prieto-Díaz&Arango fassen das Domänenmodell als Endergebnis der DA auf und definieren es sehr weitreichend. Nach Prieto-Díaz&Arango sollte das Domänenmodell folgendes enthalten:[2]

- Konzepte zur Spezifikation von Problemen und Systemen in dieser Domäne
- Grundsätze für die Spezifikationskonzepte
- Grundsätze für die Implementationspläne
- Relationen zwischen den Spezifikationskonzepten und Implementationsplänen
- Anweisungen zur Umsetzung dieser Spezifikationen in Code
- Entscheidungen, Alternativen, Einschränkungen, Erklärungen

Unser Ansatz geht nicht so weit. Wir verlangen von einem Domänenmodell:

- eine nachvollziehbare Abgrenzung der Domäne,
- eine Darstellung der Elemente und Beziehungen in dieser Domäne,
- die Erklärung der Fachsprache,
- Entscheidungen, Alternativen, Einschränkungen, Erklärungen.

Die „Darstellung der Elemente und Beziehungen“ bildet den Kern unseres Domänenmodells. Wir nennen diese Darstellung in Zukunft *Kernmodell*. Ohne die anderen Punkte wird dieses aber wertlos, da durch diese zusätzlichen Informationen der Prozeß der DA nachvollziehbar bleibt. So ist direkt einzusehen, daß das Kernmodell ohne Kenntnis der Fachsprache, die in der jeweiligen Domäne gesprochen wird, nicht verständlich ist. Die anderen Punkte haben somit den Status einer Dokumentation des Kernmodells. Warum wir diesen Ansatz gewählt haben, wird in Kapitel 2.3 näher erklärt.

[1] zitiert nach: Prieto-Díaz, Arango (a.a.O.), S. 13
[2] vgl. Prieto-Díaz, Arango (a.a.O.), S. 13 und 19

Das Domänenmodell dient als[1]

- einheitliche Referenz
- Ablage für vereinheitlichte Informationen zur Kommunikation
- Basis für die Strukturierung der wiederverwendbaren Elemente

Für die Darstellung eines Domänenmodells bieten sich drei grundlegende Möglichkeiten an:[2]

- Textuelle Modelle
- Graphische Modelle
- Ausführbare Modelle

Textuelle Modelle repräsentieren das Modell durch Prosatext oder mathematische Formeln. Diese Darstellungsart ist gut geeignet für Detailerklärungen und ein erstes Herangehen an neue Zusammenhänge, gibt aber keinen schnellen Überblick. Hier wird das graphische Modell bevorzugt, welches dem visuell orientierten Menschen sehr gut Strukturen vermitteln kann. Allerdings verlangt das graphische Modell Vorwissen über die Bedeutung der verwendeten Syntax. Textuelle und graphische Modelle ergänzen sich häufig sehr gut gegenseitig. Ausführbare Modelle sind bei der DA selten. Diese Darstellungsart erfordert einen hohen Zeitaufwand und meistens spezielle Modellierungssoftware. Alle Darstellungsarten stoßen bei großen, komplexen Systemen an ihre Grenzen, da sie dann für den Menschen nicht mehr überschaubar sind.

2.3 Einführung in die DA

Der Begriff der DA wurde von Neighbors eingeführt, um „die Aktivität des Identifizierens von Objekten und Operationen einer Klasse von ähnlichen Systemen in einer speziellen Domäne“[3] zu bezeichnen.
Anders gesagt ist Domänenanalyse der Prozeß des Identifizierens und Organisierens von Informationen über eine Domäne, um die Beschreibung dieser Domäne zu unterstützen. Wichtig wird dabei in unserem Ansatz vor allem, daß DA die gemeinsamen Elemente **mehrerer** Anwendungsbeispiele aus einer Domäne finden soll. So wird die Wiederverwendbarkeit gefördert, allerdings auch sofort die Frage nach der Machbarkeit eines solchen Prozesses aufgeworfen. Die folgenden Kapitel sollen diese Frage beantworten helfen.

Die DA ist auf zwei Annahmen gegründet:[4]

1. Die wiederzuverwendende Information ist domänenspezifisch. Die Eigenschaft der Wiederverwendbarkeit einer Information ist nicht informationsspezifisch, sie ergibt sich vielmehr aus den Beziehungen, die die Information innerhalb der Domäne hat.
2. Domänen sind stabil und zusammenhängend. Das Konzept der Wiederverwendbarkeit ist überhaupt nur anwendbar, falls die Domäne über längere Zeit stabil bleibt und alle Elemente der Domäne miteinander in Beziehung stehen.

[1] vgl. Prieto-Díaz, Arango (a.a.O.), S. 13
[2] vgl. Berard (a.a.O.), S. 189
[3] zitiert nach Jim Neighbors: Software Construction Using Components, University of California 1981
[4] vgl. Prieto-Díaz, Arango (a.a.O.), S. 10

Wie schon in 2.2.3 gesagt, wird der Begriff DA sehr unterschiedlich aufgefaßt. Berard spricht bei der DA von folgenden Schritten:[1]

- Definition der Domäne
- Sammlung einer repräsentativen Probe von Elementen der Domäne
- Analyse dieser Probe zur Identifizierung von wiederverwendbaren Elementen
- Definition von Richtlinien zur Wiederverwendung dieser Elemente
- Demonstration der Wiederverwendung dieser Elemente mit den Richtlinien
- Erstellen von Empfehlungen

Bemerkenswert ist bei dieser Darstellung, daß hier eine zeitlich Abfolge besteht und sogar die Erstellung eines Beispiels gefordert wird.

Für Prieto-Díaz&Arango ist das Domänenmodell zwar das Endergebnis der DA, aber durch die sehr umfassende Definition des Domänenmodells (siehe 2.2.3) wird die DA, genau wie bei Berard, mit einem zu weitreichenden Einsatzbereich bedacht. Nach unserer Auffassung beschränkt sich die DA auf die Untersuchung der Domäne und die Erstellung des Domänenmodells (Kernmodell + Dokumentation). Eine Codierung, welcher Art auch immer, liegt nicht im Aufgabenfeld der DA.
Wir nennen die differenzierbaren Aktionen während der DA *Aktivitäten*. Die einzelnen Aktivitäten werden in Kapitel 2.5 näher ausgeführt, aber hier schon einmal eine kurze Auflistung:

- Abgrenzung der Domäne
- Erlernen der Fachsprache / des Fachwissens
- Auswahl repräsentativer Beispiele aus der Domäne
- Analyse der Beispiele
- Erstellung des Domänenmodells

Man kann unserer Meinung nach nicht – wie Berard – von Schritten innerhalb der DA sprechen. Es ist unmöglich, eine zeitliche Abfolge von einzelnen Aktivitäten anzugeben: Die Abgrenzung der Domäne ist nur möglich, wenn repräsentative Beispiele analysiert werden. Um diese Beispiele auswählen zu können, müßte die Domäne aber schon bekannt sein. Eine DA-Gruppe kann dieses Problem meistern, indem sie die nötigen Schritte bei Bedarf zyklisch wiederholt, um so z.B. eine vorläufige Abgrenzung der Domäne auszudifferenzieren.
Mit Phasenmodellen (z.B. dem Wasserfall-Modell) ist der Prozeß der DA nicht beschreibbar; STEPS[2] ist hier besser geeignet, da je nach Bedarf die einzelnen Aktivitäten parallel oder sequentiell bearbeitet und zyklisch wiederholt werden können. Deshalb führen wir die OODA bei unserer Fallstudie mit STEPS durch, was sich sowohl durch das Projektseminar „Ein objektorientiertes Entwurfsschema für STEPS“ als Rahmen für unsere Fallstudie als auch durch den evolutionären Charakter von STEPS anbietet. Welche Konzepte von STEPS sich gut und welche sich weniger gut bei unserer Fallstudie bewährt haben, wird in späteren Kapiteln behandelt.

Um den Ablauf der Aktivitäten zu organisieren, bedarf es viel Erfahrung und Konzepten aus dem Projektmanagement. Projektetablierung, Planung, Projektbewertung, Referenzlinien, Benutzerbeteiligung, Krisenmanagement (Firefighting) und Lernstrategien sind nur einige Schlagworte aus dem Bereich Projektmanagement, die auszuführen den Rahmen dieser Einführung sprengen würde.[3, 1]

---

[1] vgl. Berard (a.a.O.), S. 187ff
[2] vgl. Floyd (a.a.O.)
[3] vgl. Niels Erik Andersen: Professional Systems Development, New York u.a. 1990

Eine Herausforderung bei der DA liegt im Verstehen der Domäne. Erst wenn das möglich ist, kann das Domänenmodell erstellt werden. Dieser Vorgang ist auch der aufwendigste und wird von Thompson&Clancey das „acquisition bottleneck“[2] genannt. Um diesen Flaschenhals überhaupt passieren zu können, ist die Befragung von *Domänenexperten*[3] unbedingt erforderlich. Nur sie können die Informationen liefern, die für eine gute Analyse nötig sind. Die Bedeutung der Domänenexperten geht allerdings weit über die Rolle eines bloßen Informationslieferanten hinaus. Sie sind gleichwertige Kommunikationspartner im DA-Prozeß.

Aus der obigen Auflistung der Aktivitäten wird klar, daß DA immer einen Lernprozeß darstellt. Allein das Erlernen der Fachsprache der Domänenexperten kann sich als äußerst schwierig erweisen. Soll z.B. die Domäne „Gebäudestatiken“ untersucht werden, ist ein Dickicht an Fachbegriffen und DIN-Normen zu lichten. Aus diesem Problem haben sich zwei antagonistische Ansätze für die Besetzung der DA-Gruppen gebildet:

1. Eine Gruppe von Domänenexperten wird in die DA eingewiesen. Diese brauchen die Fachsprache ja nicht mehr zu erlernen. Die Domänenexperten erstellen dann allein das Domänenmodell.
2. Eine Gruppe von Domänenanalytikern erlernt die Fachsprache. Die Domänenanalytiker erstellen dann im Dialog mit Domänenexperten das Domänenmodell.

Da meistens die Domänenexperten am wenigsten verfügbar sind, wird mit Ansatz zwei diejenige Lösung gewählt, welche die Domänenexperten am wenigsten belastet. Zudem ist der Ausbildungsstand der Domänenexperten in Bezug auf DA häufig dergestalt, daß Ansatz eins nicht zu realisieren ist.
Es kann allerdings auch vorkommen, daß die *funktionellen Rollen*[4] Domänenexperte und Domänenanalytiker in einer Person zusammenfallen. So ist ein Domänenanalytiker für die Domäne „Textverarbeitungssysteme“ häufig auch ein Domänenexperte. Eine solche Konstellation birgt die Gefahr, daß zu wenig externe Informationen in den DA-Prozeß einfließen und das Endergebnis zu stark auf die alleinigen Bedürfnisse des Domänenanalytikers/Domänenexperten ausgerichtet ist.

2.3.1 Abgrenzung zu anderen Bereichen

DA liegt an einer Nahtstelle von vielen Disziplinen: System- und Anforderungsanalyse, Spezifikation und Implementation von Software, Informationserwerb und -repräsentation.[5]

Betrachtet man die DA z.B. im Vergleich mit der System- und Anforderungsanalyse, so erkennt man, daß die DA die Komponenten der System- und Anforderungsanalyse auf einer Metaebene enthält:
Die Systemanalyse beschäftigt sich mit speziellen Aktionen und Elementen in speziellen Systemen (bei der Erstellung des Vorlesungsverzeichnisses z.B. die Aktionen und Elemente der Raumvergabe der Universität Hamburg), wohingegen die DA Aktionen und Elemente in allen

[1] vgl. Niels Fricke, Christian Fürll, Holger Koschek, Thomas Pfohe, Peter v. Savigny: Systementwicklung - Projektmanagement, Projekt Softwaretechnik 1995
[2] vgl. T. F. Thompson, W. J. Clancey: A Qualitative Modeling Shell for Process Diagnosis, IEEE Software 1986
[3] Ein Domänenexperte ist ein Mensch, dessen Arbeitsbereich in der Domäne liegt. So ist z.B. ein Bankangestellter für die Domäne „Bankgeschäfte“ ein Domänenexperte.
[4] vgl. Floyd (a.a.O.), S 62ff
[5] vgl. Prieto-Díaz, Arango (a.a.O.), S. 11

Systemen eines Anwendungsbereiches (hier also die gemeinsamen Aktionen und Elemente bei der Erstellung des Vorlesungsverzeichnisses an allen Universitäten in Deutschland) betrachtet.
Die Anforderungsanalyse beschäftigt sich mit der Identifizierung, Akquisition und Repräsentation von Informationen für ein Problem, wohingegen die DA die Identifizierung, Akquisition und Repräsentation von Informationen für Klassen von Problemen betrachtet.

Die Spezifikation und Implementation von Software wird vom Domänenmodell als Ergebnis der DA unterstützt, besonders in Hinsicht auf die Wiederverwendbarkeit.

Da die gesamte DA damit zu tun hat, Informationen über die Domäne zu erwerben und zu organisieren, besteht eine starke Verwandtschaft zu Informationserwerb und -repräsentation.

Viele der in der DA verwendeten Techniken werden auch in den oben genannten Disziplinen eingesetzt.
Als Erweiterung zum Software engineering schlagen Prieto-Díaz&Arango den Begriff *Domain engineering* vor. Dieser steht für den Prozeß von der Aufgabenstellung bis zur Wartung der gesamten in einer Organisation für die Wiederverwendung aktueller und zukünftiger Elemente nötigen *Infrastruktur* (Programme, Dokumentation, Personal, usw.). [1]

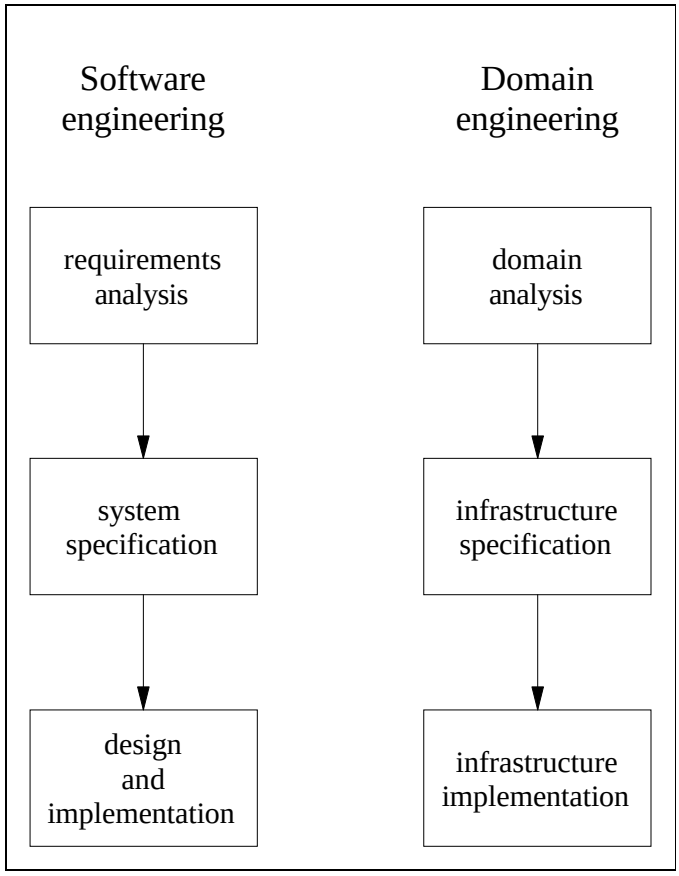

Abbildung 2.3-1: Vergleich Software engineering mit Domain engineering[2]

[1] vgl. Prieto-Díaz, Arango (a.a.O.), S. 15
[2] vgl. Prieto-Díaz, Arango (a.a.O.), S. 15

Während also das Software engineering die Bereiche Anforderungsanalyse, Systemspezifikation, Design und Implementation enthält, geht das Domain engineering auf die Ebene der Domänen, indem es DA, Infrastrukturspezifikation[1] und Infrastrukturimplementation[2] abdeckt (siehe Abbildung 2.3-1).

2.3.2 Klassifizierung

DA kann nach Art der gewählten Systeme klassifiziert werden:[3]

- *Horizontale DA*:
  Untersucht wird eine Domäne, deren Anwendungen in größeren Anwendungssystemen vorkommen können. Wird z.B. die Domäne „Textverarbeitungen" untersucht, so können Textverarbeitungen innerhalb der Domäne „Erstellung eines Vorlesungsverzeichnisses" benutzt werden. Bei der horizontalen DA werden somit eher technische Systeme untersucht.
- *Vertikale DA*:
  Untersucht wird eine Domäne, für die ähnliche Anwendungen konstruiert werden sollen. Wird z.B. die Domäne „Bank" untersucht, so ergibt sich für jede individuelle Bank eine Variation des Banksystems. Bei der vertikalen DA werden somit eher anwendungsspezifische Systeme untersucht.

Eine andere Möglichkeit ist die Klassifizierung nach der Art der wiederzuverwendenden Elemente:[4]

- *Funktionale DA* (FDA)
  sucht Wiederverwendbares im Umfeld von Funktionen, also Funktionen, Unterprogramme, Prozeduren und Programme.
- *Objektorientierte DA* (OODA)
  sucht Wiederverwendbares im Umfeld von Elementen, also Objekte und Klassen.

2.4 Informationsquellen

Ein wesentlicher Aspekt der DA ist die Quantität und Qualität der zur Verfügung stehenden Informationsquellen.
Es folgt eine (sicherlich unvollständige) Liste der möglichen Informationsquellen mit einer kurzen Abwägung der jeweiligen Vor- und Nachteile.[5] Generell gilt: Alles, was an Informationen zu einer Domäne zur Verfügung steht, sollte zumindest gesichtet werden, auch wenn aus Zeit- oder Geldgründen nicht alles bearbeitet werden kann:

[1] Die Infrastrukturspezifikation beschreibt den Vorgang der Spezifikation der nötigen Infrastruktur für die wiederverwendbaren Elemente.
[2] Die Infrastrukturimplementation beinhaltet die Implementation der gefundenen Spezifikationen.
[3] vgl. Booch, G.: Object-Oriented Design with Applications, Benjamin/Cummings, Redwood City, CA 1991, S. 142
[4] vgl. Berard (a.a.O.), S. 185
[5] vgl. Prieto-Díaz, Arango (a.a.O.), S. 18f

- Aussagen von Domänenexperten
  Domänenexperten können hervorragend einen Überblick über die Domäne geben. Ihre Aussagen erlauben eine Strukturierung der Informationsvielfalt. Auch können Domänenexperten existierende Zustände oder Handlungen begründen und Erfahrungswerte weitergeben, welche meistens nirgendwo niedergelegt sind. Im Gegensatz zu textuellen Aufzeichnungen können sie gefragt werden, also flexibel antworten.
  Domänenexperten stehen nur begrenzt zur Verfügung, da deren Zeit meistens knapp und teuer ist. Informationsbeschaffung von Domänenexperten ist schwierig und langwierig, da sie eine eigene domänenspezifische Fachsprache sprechen und durch ihr Vorwissen unbewußte Annahmen in ihre Antworten einflechten.
- Technische Literatur
  Technische Literatur liefert präzise und detaillierte Informationen, enthält aber selten Erklärungen oder historische Begründungen.
- Existierende Anwendungen
  Existierende Anwendungen können Fragen an die Domänenexperten auslösen und als Positiv- oder Negativbeispiele dienen.
  Die Auswahl der in Frage kommenden Anwendungen kann schon zu falschen Ergebnissen führen: Da existierende Anwendungen aus einer großen Zeitspanne, von unterschiedlichsten Softwaretechnik-Schulen, Betriebs- und Computersystemen stammen können, ist die Abstraktion von existierenden Anwendungen sehr schwierig. Die eben genannten Vorbedingungen erfordern eine genaue Abwägung des Gültigkeitsbereiches einer Abstraktion, die sonst häufig in die falsche Richtung führen kann.
- Aufzeichnungen über die Entwicklung der Domäne
  Aufzeichnungen über die Entwicklung der Domäne geben darüber Aufschluß, ob ein Element der Domäne eher dynamisch oder eher statisch ist: Die DA ist u.a. auf der Annahme gegründet, daß Domänen stabil sind (siehe 2.3). Das bedeutet allerdings nicht, daß sich ein Element der Domäne nicht in der Zeit verändern kann. Solche Elemente sind schwieriger in das Domänenmodell einzufügen und erfordern mehr Ressourcen[1] und Arbeitskräfte für deren Analyse. Somit läßt sich durch Aufzeichnungen über die Entwicklung der Domäne der Einsatz von Ressourcen und Arbeitskräften verbessern, was Engpässe verhindern hilft.
  Allerdings sind solche Aufzeichnungen – wenn überhaupt vorhanden – häufig unvollständig und Interpretationen infolgedessen meistens unzutreffend.

2.5 Aktivitäten der DA

Im folgenden werden die Aktivitäten (differenzierbare Aktionen) unseres DA-Ansatzes beschrieben. Die fünf im folgenden näher ausgeführten Aktivitäten haben wir durch Literaturstudium[2,3,4] und anschließenden Abgleich mit unserer OODA am Beispiel „Erstellung eines Vorlesungsverzeichnisses“ erarbeitet. Wir weisen noch einmal darauf hin, daß **kein** zeitlicher Ablauf zwischen den einzelnen Aktivitäten vorgegeben wird. Des weiteren sind die Aktivitäten so stark gegenseitig abhängig, daß sie in der Praxis stark verwoben ausgeführt werden. Die ge-

[1] Ressourcen bezeichnen hier keine Arbeitskräfte, sondern Computer, Zeit, Papier, Geld, Material, Gebäude usw.
[2] vgl. Berard (a.a.O.), S. 187f
[3] vgl. Prieto-Díaz, Arango (a.a.O.), S. 19
[4] vgl. Prieto-Díaz: Domain Analysis for Reusability, in: Will Tracz: Software Reuse: Emerging Technology, IEEE 1987, S.348ff

nauen Beziehungen zwischen den einzelnen Aktivitäten, die zum fertigen Domänenmodell führen, sind schwer zu klassifizieren. Da es keine Taxonomien (Klassifizierungsschemata) der Beziehungen gibt, hoffen wir, daß unsere Beschreibung eines DA-Prozesses in den folgenden Kapiteln wenigstens einen ersten Eindruck der DA vermittelt. Allerdings kann nur die eigene Durchführung einer DA eine tiefere Einsicht schaffen.

2.5.1 Abgrenzung der Domäne

Eine der ersten Aufgaben ist die Festlegung der *Domänengrenzen*, also des bei der Analyse betrachteten Ausschnittes des Gesamtsystems. Für den hier verfolgten Ansatz ist schon die Frage, wie weit man die Grenzen für die zu analysierenden Systeme steckt, in keinster Weise trivial. Die von uns in den folgenden Kapiteln verwendete Domäne „Erstellung eines Vorlesungsverzeichnisses“ kann man z.B. rein geographisch beschränken, in der Annahme, daß eine Einbeziehung der amerikanischen Abläufe weniger Übereinstimmungen erzielt als ein alleiniger Vergleich der vorliegenden Systeme in Deutschland. Dafür werden bei internationalem Vergleich die erkannten Elemente allgemeiner sein, was sie für einen größeren Anwendungsbereich einsetzbar macht. Eine zu weitgehende Verallgemeinerung erzeugt allerdings Elemente, die zu wenig domänenspezifische Informationen tragen. Hier besteht die Aufgabe eines Domänenanalytikers darin, einen Mittelweg zu finden. Eine weitere Frage ergibt sich durch die Überlegung, welche Einrichtungen ein Vorlesungsverzeichnis erstellen müssen. Typischerweise sollten dies die Universitäten sein, aber auch andere Einrichtungen (z.B. Fachhochschulen oder Firmen) sind vorstellbar. Für die Entscheidung, welche Einrichtungen ausgewählt werden, ist eine genaue Klärung des Begriffes „Vorlesungsverzeichnis“ vonnöten. Genauso wie die geographische Auswahl unvermeidbare Einschränkungen mit sich bringt, tut dies auch die Auffassung des Begriffs „Vorlesungsverzeichnis“. Hier wird wieder die Verzahnung der einzelnen Schritte deutlich, denn wie soll sich die DA-Gruppe über den Begriff „Vorlesungsverzeichnis“ klar werden, ohne diesen in repräsentativen Beispielen analysiert oder sich die Fachsprache angeeignet zu haben.

2.5.2 Erlernen der Fachsprache / des Fachwissens

Das Erlernen der Fachsprache ist immer dann nötig, wenn der Domänenanalytiker kein Domänenexperte ist. Ein Erlernen der Fachsprache ist eng mit dem Erwerb des dazugehörigen Fachwissens verbunden; die einzelnen Fachwörter bleiben sonst leere Worthülsen. Dieser Lernprozeß wird in den meisten Fällen nötig sein. Am besten geeignet ist auch hier wieder ein Gespräch mit einem Domänenexperten, allerdings helfen ebenfalls technische Unterlagen weiter. Die erlernten Begriffe der Fachsprache müssen in das Domänenmodell aufgenommen werden, damit ein gemeinsames Verständnis in der DA-Gruppe geschaffen werden kann.
Ohne eine zumindest grundlegende Kenntnis der Fachsprache wird es zu Mißverständnissen innerhalb der Gruppe, aber auch mit den Domänenexperten kommen. Ein weiterer Grund besteht in der beabsichtigten Ausrichtung der zu konstruierenden Software an die Sprache der Anwender,[1] wie sie auch schon von STEPS[2] gefordert wird.

2.5.3 Auswahl repräsentativer Beispiele aus der Domäne

Ebenfalls unter die ersten Aufgaben fällt die Wahl repräsentativer Beispiele. Selten wird genug Zeit und Geld vorhanden sein, um alle Systeme der Domäne zu analysieren. Hat die DA-Gruppe z.B. Deutschland als geographischen Bereich für die Domäne „Erstellung von Vorlesungsverzeichnissen“ gewählt, so werden wohl kaum alle Einrichtungen analysiert werden

[1] vgl. Klaus Kilberth, Guido Gryczan, Heinz Züllighoven: Objektorientierte Anwendungsentwicklung, Braunschweig, Wiesbaden 1994, S. 92ff
[2] vgl. Floyd (a.a.O.)

können, die ein Vorlesungsverzeichnis erstellen. Vielmehr werden einige exemplarisch ausgewählt.
Dieser Auswahlprozeß verlangt wiederum eine oberflächliche Analyse der Kandidaten, um möglichst viele Unterschiede in der VV-Erstellung mit möglichst wenigen Beispielen abdecken zu können.

2.5.4 Analyse der Beispiele

Mit der Analyse beginnt die Abstrahierung und Klassifizierung der Elemente in der Domäne. Diese Tätigkeiten laufen häufig parallel, zusätzlich wird dabei die Grenze der Domäne immer schärfer.
Die erkannten Elemente müssen in eine Nomenklatur (Begriffsschema) eingebettet werden, die als Basis die Fachsprache der Domäne hat. Existierende fachliche Begriffe müssen den Elementen also zugeordnet werden. Hier ergibt sich ein Problem: Wie ordnet man Begriffe ein, die zwar gleich lauten, aber innerhalb der Domäne eine unterschiedliche Bedeutung haben (Homonyme)? Neue Begriffe einzuführen ist nicht der richtige Weg, da man sich so von der Fachsprache entfernt. Die unterschiedlichen Bedeutungen der Begriffe müssen vielmehr den funktionellen Rollen zugeordnet werden, die sie benutzen. Für die einzelnen funktionellen Rollen ist der Begriff eindeutig, nur der Domänenanalytiker muß durch entsprechende Dokumentation die Begriffe klar voneinander trennen.
Die Elemente müssen auch in eine Taxonomie (Klassifizierungsschema) eingefügt werden, um Hierarchien von Elementen bilden zu können, da es meistens zu viele Elemente geben wird, um sie in einer flachen Anordnung noch überblicken zu können.
Ein exemplarischer Analyseprozeß wird am Beispiel der Domäne „Erstellung eines Vorlesungsverzeichnisses“ ab Kapitel 3 dargestellt. Denn genau wie Prieto-Díaz&Arango schon bemerkt haben: der eigentliche Prozeß der DA ist immer noch ungeklärt.[1] Aus diesem Grund findet sich bei Prieto-Díaz&Arango keine nähere Beschreibung dieses Prozesses.

Die Abstrahierung und Klassifizierung der Elemente reicht für ein Domänenmodell im Normalfall nicht aus. Die Zusammenhänge und Strukturen zwischen den Elementen fließen als Strukturwissen mit in das Kernmodell ein.
Die Nomenklaturen und Taxonomien sollten sich aus diesem Strukturwissen ergeben. Das ist die natürlichste – aber nicht einfach zu realisierende – Art der Anordnung von Elementen: in der Struktur der Domäne.

Alle diese Tätigkeiten werden immer wieder durchlaufen und erfordern immer neue Informationsbeschaffung.

2.5.5 Erstellung des Domänenmodells

Die Erstellung des Domänenmodells ist wieder ein nebenläufiger Prozeß, der von den Ergebnissen der Analyse der Beispiele lebt. Mit der Erstellung des Kernmodells als meistens graphisches Modell wird auch die entsprechende Dokumentation erstellt; diese beiden Komponenten zusammen machen das Domänenmodell aus. Besonders wichtig ist es, Entscheidungen und Alternativen zu motivieren, die zeigen, warum das Kernmodell von der DA-Gruppe gerade so entworfen und nicht ein anderer, vielleicht näherliegender, Ansatz gewählt wurde. Die DA bleibt so auch für andere DA-Gruppen nachvollziehbar.

[1] vgl. Prieto-Díaz, Arango (a.a.O.), S. 20

2.6 Ergebnis der DA

Das Ergebnis der DA ist das Domänenmodell. Das Domänenmodell wird als Basis für den Entwurf von wiederverwendbarer Software benutzt, da hier die wiederverwendbaren Elemente und Strukturen einer ganzen Domäne abgebildet sind. Auch Aspekte von Analyse und Design sind im Domänenmodell vorhanden, sind also bei der Softwareentwicklung schnell verfügbar. Will man für die entsprechende Domäne Software entwickeln, so hat man mit dem Domänenmodell eine weitreichende Basis, die durch die Dokumentationen, das schon gesammelte Detail- und Strukturwissen, die getroffenen Entwurfsentscheidungen und die möglichen Alternativen gebildet wird. Ist bereits wiederverwendbare Software für diese Domäne auf Basis des Domänenmodells entwickelt worden, so sind wiederverwendbare Komponenten vorhanden und das Domänenmodell kann als eine Art Index für diese Komponenten genutzt werden.

2.7 Der Domänenanalytiker

Da der gesamte DA-Prozeß zumindest leitend von einem Domänenanalytiker durchgeführt wird, ist eine Auflistung seiner Aufgaben und der von ihm geforderten Fähigkeiten recht aufschlußreich.[1,2]

Folgende Punkte fallen in den Aufgabenbereich eines Domänenanalytikers:

- Projektmanagement
- Informationsbeschaffung
- Informationsorganisation
- Analyse, Abstraktion und Klassifizierung von Elementen und Beziehungen
- Modellentwicklung
- Verifikation der Korrektheit des Modells
- Validierung des Modells gegen existierende Systeme
- Analyse der Effekte von Veränderungen auf das Modell

Diese Fähigkeiten sollte ein Domänenanalytiker besitzen:

- Lernfähigkeit
- Abstraktionsfähigkeit
- Korrelationsfähigkeit
- Fähigkeit zum Entwickeln von Analogien
- Kommunikationsfähigkeit

Diese Wissensbereiche sollte ein Domänenanalytiker abdecken können:

- Konzepte und Methoden des Projektmanagements
- Konzepte und Methoden der Wiederverwendbarkeit
- Konzepte und Methoden der Objektorientierung (bei OODA)
- Konzepte und Methoden der Funktionsorientierung (bei FDA)
- Kenntnisse über die Domäne (Fachwissen)

Bei alledem ist Erfahrung aus vorhergehenden Projekten für einen Domänenanalytiker eine große Hilfe.

[1] vgl. Prieto-Díaz, Arango (a.a.O.), S.21
[2] vgl. Berard (a.a.O.), S.187

Bei dieser Menge von Anforderungen an einen Domänenanalytiker wird eines klar: DA ist eine Gruppenarbeit.

2.8 Schlußfolgerungen

Die DA soll Modelle für die Schaffung, Erfassung und Darstellung von Detail- und Strukturwissen in Domänen liefern, um so den Vorgang der Konstruktion von wiederverwendbarer Software zu unterstützen. Eine Codierung von Software ist kein Bestandteil der DA.

Durch den Aktivitätscharakter der DA sind Phasenmodelle (Wasserfall-Modell) nicht geeignet, einen DA-Prozeß angemessen zu unterstützen.

Die Kommunikation zwischen den Domänenanalytikern und den Domänenexperten ist ein integraler Bestandteil unseres DA-Konzeptes. Sie ist eine Voraussetzung für ein verwendbares Domänenmodell.

Die DA betont das Wiederverwenden von Analyse und Design. Auch dadurch wird der Vorgang der Softwarekonstruktion für eine Domäne durch eine wiederverwendbare Analyse- und Designstudie erleichtert, die nach Erstellung eines Domänenmodells vorhanden ist.

Anhand des Falles eines schon vorliegenden Domänenmodells kann man eine wichtige Eigenschaft der DA motivieren: Die DA ist ein nie endender Prozeß. In Kapitel 2.5.2 haben wir schon darauf hingewiesen, daß jede DA-Gruppe aus Ressourcengründen niemals alle möglichen Beispiele einer Domäne zur Analyse heranziehen kann. Je mehr Beispiele es sind und je breiter deren Anwendungsbereich ist, um so allgemeiner (mit den entsprechenden Vor- und Nachteilen) werden die im Domänenmodell abgebildeten Elemente und Strukturen sein. Wenn jetzt ein neues Beispiel analysiert wird, kann sich herausstellen, daß dieses Beispiel veränderte oder grundsätzlich unterschiedliche Elemente oder Strukturen besitzt, obwohl es offensichtlich aus der Domäne stammt. Ob das bestehende Domänenmodell geändert, erweitert oder so belassen wird, ist eine Entscheidung des Projektmanagements, aber die DA wird solange ausgeführt, wie neue Software für diese Domäne erstellt wird oder alte Software gewartet werden muß.

Obwohl die erhöhte Wiederverwendbarkeit als Hauptzweck der DA gesehen wird, ergeben sich andere sehr positive Nebeneffekte. Für die Domänenexperten zeigen sich durch den Analyseprozeß neue Einsichten in ihren Arbeitsbereich und oft werden verborgene Abläufe, die bisher unreflektiert durchgeführt wurden, als **begründbar** falsch oder richtig erkannt. Allein die Optimierungsmöglichkeiten in Verwaltung und Organisation, die sich durch die Erstellung des Modells und dem daraus folgenden Detail- und Strukturwissensgewinn ergeben, sind fast in jedem Fall für sich genommen eine DA wert. Daraus läßt sich die Schlußfolgerung ableiten, daß DA mehr ist als eine auf die Informatik beschränkte Methode zur Unterstützung der Softwareentwicklung. DA läßt sich – in Analogie zur Systemanalyse – generell für die übergreifende Erstellung von Modellen benutzen. Somit ist das Konzept der DA nicht allein auf die Softwareerstellung beschränkt.

DA ist eine informationsintensive Tätigkeit, und die Methoden und Formalisierungen für den Übergang von Informationen in das Kernmodell befinden sich noch in der Entwicklung.
Das Hauptproblem der DA liegt in der enormen Größe und Komplexität von Detail- als auch Strukturwissen, die selbst einfachen Domänen innewohnt. Leider wird durch die unvermeidbar kleinen und vereinfachten Beispiele in der Literatur der Eindruck erweckt, die Komplexität von Domänen sei nicht so groß.

3 Erschließung des Vorgangs der VV-Erstellung

Im folgenden kommen wir nun zur praktischen Umsetzung der Domänenmodellierung anhand des im Projektseminars vorgegebenen Fallbeispiels der VV-Erstellung.
Ein Ziel unserer Gruppe war also unter anderem die Erstellung eines Domänenmodells, so wie wir es verstehen.[1] Dazu gehören die nachvollziehbare Abgrenzung der Problemdomäne, das Erlernen der Fachsprache und die Darstellung der Elemente und Beziehungen in dieser Problemdomäne.
Unsere DA-Gruppe hatte somit als erste Aufgabe, sich den Anwendungsbereich der Erstellung eines Vorlesungsverzeichnisses zu erschließen. Eine Auswahl und Untersuchung von Beispielen aus dieser Problemdomäne sollte uns da weiterhelfen, denn eine Domänenmodellierung hat nur dann einen Zweck, wenn die Modellierer mehrere unterschiedliche Ausprägungen dieses Vorgangs untersucht haben, um so einen Überblick über mögliche Vorgehensweisen bei der VV-Erstellung zu erlangen. Erst damit kann eine sinnvolle Abgrenzung der Problemdomäne begründbar gemacht werden. Aus diesem Grund hat sich unsere Domänenmodellierungsgruppe zum Anfang des Projektseminars in zwei Untergruppen gegliedert: Stefan Fröhlich, Oliver Satz und Marko Fricke haben sich anhand der Vorstudie[2] über die Vorgänge bei der Erstellung des VV in Berlin informiert, und wir haben als zweiten Anwendungsbereich die VV-Erstellung am Fachbereich Informatik der Universität Hamburg untersucht. Es handelte sich hierbei also um eine vertikale Domänenanalyse[3], über deren Problematik, daß nur zwei Beispiele analysiert wurden, sich die Gruppe von vornherein bewußt war. Auf diesen Punkt werden wir noch gesondert zu sprechen kommen.
Um später dann die untersuchten Verfahren in Berlin und Hamburg vergleichbar zu machen und über sie besser diskutieren zu können, haben wir uns zunächst auf die grafische Veranschaulichung mit Hilfe von *Aufgabennetzen*[4] geeinigt, wie sie im STEPS-Kontext benutzt werden. Die Hoffnung war, anhand dieses grafischen Darstellungsmittels Unterschiede und Gemeinsamkeiten der VV-Erstellung in Berlin und Hamburg besser herausarbeiten zu können, um die Problemdomäne abzugrenzen, und um die Fachsprache für uns zugänglich zu machen. Gleichzeitig wollten wir herausfinden, inwieweit Aufgabennetze überhaupt für die DA und insbesondere für unsere angestrebte objektorientierte DA sinnvoll einsetzbar sind.
Einführend zu den Aufgabennetzen wurde das exemplarische Vorgehen beim Erstellen eines Vorlesungsverzeichnisses schriftlich in Form von *Szenarios*[5] niedergeschrieben.

3.1 IST-Zustand Hamburg

Wir wollen in diesem Abschnitt darlegen, wie wir uns dem Themengebiet der VV-Erstellung am Fachbereich Informatik der Universität Hamburg genähert haben. Dazu haben wir durch Rücksprache mit Anita Krabbel, unserer DA-Gruppenleiterin, erfahren, daß Herr Michael König[6] semesterweise an die *Arbeitsbereich-Lehrplanbeauftragten* (*AB-Lehrplanbeauftragten*) mit einem Schreiben herantritt, in dem sie aufgefordert werden, die von den Veranstaltern angebotenen *Lehrveranstaltungen* (*LV*) zu kommentieren. Diese Ausführungen fließen dann ins *kommentierte Vorlesungsverzeichnis* (*KVV*) ein. Aus diesem Grund war also Herr König unser erster Ansprechpartner und somit unser erster Domänenexperte. Von ihm erfuhren wir, daß

[1] siehe Kapitel 2.5
[2] siehe Anhang C
[3] siehe Kapitel 2.3.2
[4] vgl. Floyd:, Christiane: Arbeitsunterlagen zur Lehrveranstaltung Einführung in die Softwaretechnik, Universität Hamburg SS 93,S.75ff
[5] vgl. Kilberth et al. (a.a.O.), S.96ff
[6] Herr König ist Mitarbeiter am Informatik-Rechenzentrum der Universität Hamburg.

zusätzlich Herr Prof. Kudlek für die *Raumvergabe* und Herr Prof. Menzel für die *Lehrveranstaltungsermittlung* (LV-Ermittlung) und die *Deputatsverwaltung* zuständig ist. Unter Deputatsverwaltung sei dabei die Überprüfung auf Erfüllung der Lehrveranstaltungsstunden zu verstehen, die jeder Veranstalter pro Semester leisten muß. Alle drei Personen haben wir zu diesen Themengebieten befragt. Wichtig war für uns vor allem die Klärung folgender Fragen:

- Welche Aufgaben und Tätigkeiten gibt es und wie sieht ihr Ablauf aus?
- Wer führt sie wie aus?
- Was gibt es für Zwischenprodukte auf dem Weg zum fertigen Vorlesungsverzeichnis?
- Welche Bestimmungen bzw. Auflagen müssen beachtet werden?

Aus diesen Interviewergebnissen haben wir zunächst Szenarios erstellt, um dann darauf aufbauend Aufgabennetze herauszuarbeiten. Die Szenarios dienen zusätzlich dazu, einen dynamischen Ablauf der existierenden Tätigkeiten exemplarisch darzustellen. Dies ermöglicht dem Leser einen ersten Einblick und Überblick über das Themengebiet. Daher können die Szenarios auch gut dazu verwandt werden, die darauf aufbauenden Aufgabennetze verstehen zu helfen. Allerdings sei darauf hingewiesen, daß gerade Szenarios keinen Anspruch auf Vollständigkeit haben, sondern „sie helfen den Entwicklern und zukünftigen Anwendern, einen Lern- und Modellierungsprozeß voranzutreiben“.[1] Szenarios und Aufgabennetze sollen sich ergänzen. Die Aufgabennetze sind als grafisches Darstellungsmittel sehr wichtig, und zusätzlich kann man sich das tatsächliche Vorgehen zusammen mit den Szenarios gut verdeutlichen, indem man versucht, die Szenarios in den Netzen nachzuvollziehen.

3.1.1 Ist-Szenarios

Szenario: Ermitteln der Lehrveranstaltungen

Der *Lehrplanbeauftragte* des Fachbereichs Informatik verschickt circa ein Jahr vor Beginn des betreffenden Semesters Formulare für die Beschreibung der Lehrveranstaltungen[2] und zusätzlich Informationen zu den zu leistenden Deputaten an die einzelnen Arbeitsbereiche. Deren Lehrplanbeauftragte (AB-Lehrplanbeauftragte) sorgen dann dafür, daß alle Lehrveranstalter der einzelnen Arbeitsbereiche ihre Veranstaltungen eintragen und daß die Auflagen für die Pflichtlehre erfüllt werden. Die ausgefüllten Formulare (bzw. adäquate Lehrveranstaltungsbeschreibungen) werden dann an den Lehrplanbeauftragten zurückgegeben. Einige Arbeitsbereiche verwenden nicht diese verschickten Formulare, sondern haben eigene entwickelt, die aber denselben Informationsgehalt haben. Der Lehrplanbeauftragte prüft die Lehrveranstaltungsbeschreibungen auf Korrektheit und verwaltet die Deputate.
Die einzelnen *Lehrtableaus* der Arbeitsbereiche, als Zusammenfassung der einzelnen LV-Formulare, werden nun zu einem einzigen Lehrtableau zusammengeführt. Dieses muß nun durch den *Fachbereichsrat* (*FBR*) verabschiedet werden.

Szenario: Rechnereingabe der Lehrveranstaltungen

Der *Raumplaner* überträgt nach der Verabschiedung des Lehrtableaus durch den FBR die Daten der Lehrveranstaltungsbeschreibungen in das Stundenplanprogramm *Stupla*. Dieses Programm unterstützt ihn im wesentlichen bei der Gewährleistung von Überschneidungsfreiheit bei *Pflichtveranstaltungen* und bei der Raumverteilung. Stupla identifiziert Lehrveranstaltungen nach der *LV-Nummer*. Durch diese LV-Nummern sind auch die Pflichtveranstaltungen identifizierbar. Um Veranstaltungen wie die Fachbereichsratssitzung bei der Raumvergabe berück-

[1] vgl. Kilberth et al. (a.a.O.), S. 97
[2] zum Aufbau der Formulare siehe Anhang E und F

sichtigen zu können, wurde ein spezieller Nummernschlüssel für sonstige Veranstaltungen eingeführt.

Szenario: Raumvergabe und Abstimmung von Überschneidungen

Zunächst läßt der Raumplaner in den einzelnen Arbeitsbereichen eine informelle Raumvorplanung durchführen, die ihm als Anhaltspunkt für Raumpräferenzen dient. Die Zeiten für Räume, die nicht zum Fachbereich Informatik gehören (z.B. die Hörsäle am Campus), beantragt er bei der Universitätsverwaltung und kann dabei auch Raum- und Zeitwünsche äußern. Später bekommt er dann entsprechende Räume zugewiesen, die jedoch nicht mit den gewünschten Raum- und Zeitwünschen übereinstimmen müssen. Auf Grundlage dieser Daten wird nun die Raumvergabe mit Hilfe des Programms Stupla durchgeführt. Obwohl das Programm eine automatische Raumvergabe unterstützt, wird diese nicht genutzt, um individuellen Raumwünschen der Arbeitsbereiche besser und schneller gerecht werden zu können.
Ergebnis dieses Arbeitsschrittes ist das von Stupla generierte Vorlesungsverzeichnis,[1] in dem alle Veranstaltungen mit LV-Nummer, Titel, Veranstalter, Raum und Zeit aufgeführt sind.
Das Vorlesungsverzeichnis wird zur Kontrolle an die Lehrplanbeauftragten der einzelnen Arbeitsbereiche verschickt. Eventuelle Fehler im VV oder weitere Wünsche der Lehrveranstalter werden mit dem Raumplaner über die AB-Lehrplanbeauftragten direkt abgestimmt und die Änderungen mit Hilfe von Stupla eingearbeitet.

Szenario: Erstellen des kommentierten Vorlesungsverzeichnisses

Der *KVV-Planer* verschickt circa 4 Wochen vor Drucklegung ein Anschreiben und eine Erklärung zu den Druckformatvorlagen[2] über die AB-Lehrplanbeauftragten an die einzelnen Lehrveranstalter. Diese liefern eine Lehrveranstaltungsankündigung mit Kommentar als formatierten MS-Word-Text an den KVV-Planer, der die Texte sammelt, korrigiert und einen Layoutabgleich durchführt. Dies geschieht in Absprache mit den Lehrveranstaltern und nachfolgender Benachrichtigung des Raumplaners zur Aktualisierung des Stupla-Datenbestandes, falls sich die Korrekturen auch auf die von Stupla geführten Daten beziehen.
Nach einer erneuten Verabschiedung des Lehrtableaus durch den Fachbereichsrat wird das Vorlesungsverzeichnis für den Abdruck im Vorlesungsverzeichnis der Universität Hamburg freigegeben. Auch das KVV kann jetzt in Druck gehen.

3.1.2 Darstellung durch Aufgabennetze: 1. Version

Obige Szenarios haben wir in einer ersten Version in mehrere Aufgabennetze überführt. Dazu wurde von uns ein Übersichtsnetz und insgesamt sieben Verfeinerungen erstellt. Diese werden nun folgend dargestellt und sind grundlegend aus den oben beschriebenen Szenarios entwickelt worden. Bei auftretenden Fragen haben wir uns zusätzlich an die entsprechenden Domänenexperten gewandt.

[1] siehe Anhang D
[2] siehe Anhang E

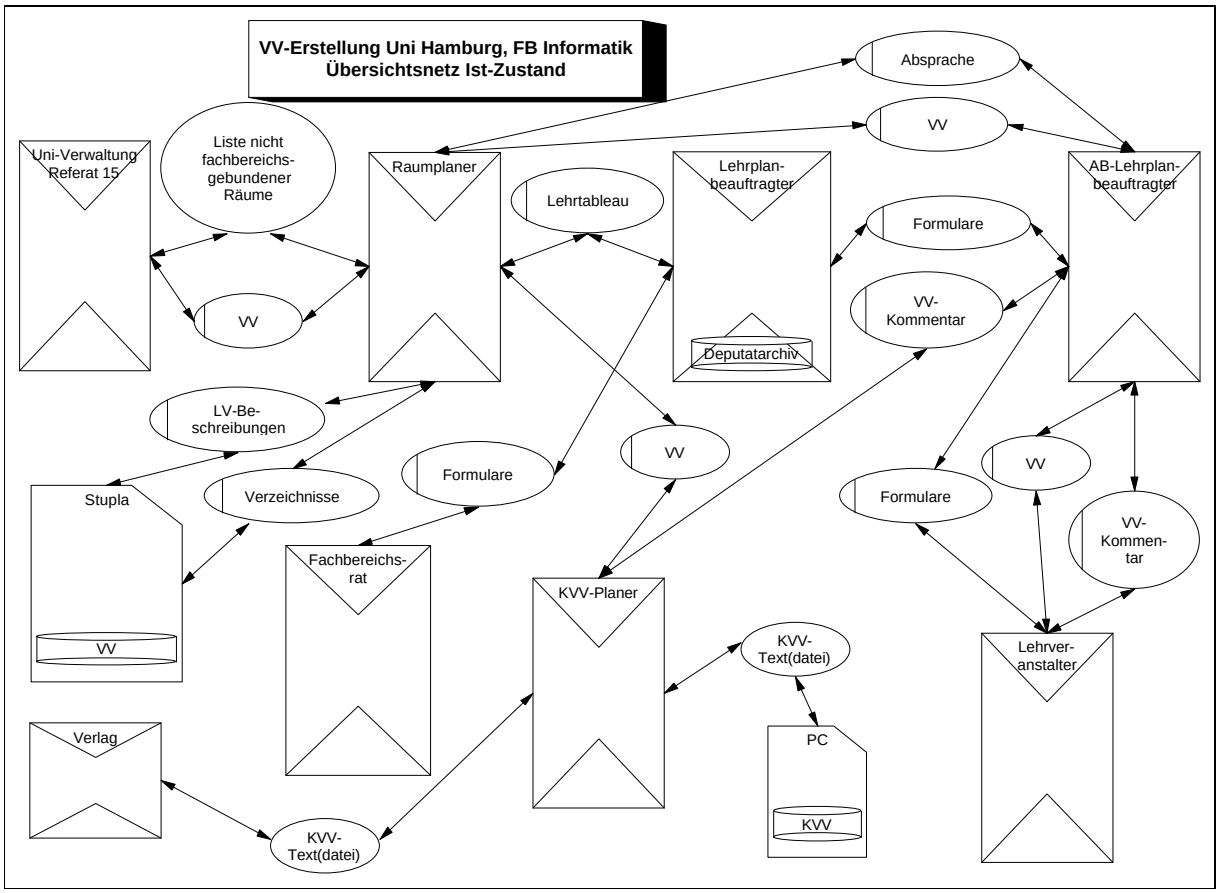

Abbildung 3.1-1: Übersichtsnetz Hamburg

Zu jeder Person, die am Fachbereich Informatik (FB Informatik) mit der Erstellung des Vorlesungsverzeichnisses zentral betraut ist, wurde von uns eine funktionelle Rolle eingeführt (Lehrplanbeauftragter, Raumplaner und KVV-Planer). Dadurch wäre eine zusätzliche Einbeziehung von Tätigkeiten in dieses Übersichtsnetz unserer Meinung nach zu unübersichtlich geworden; diese wurde deshalb in die erste Verfeinerungsstufe verlegt. Nachfolgend werden deshalb diese drei funktionellen Rollen einzeln weiter verfeinert, während die anderen funktionellen Rollen hier nicht weiter betrachtet werden.

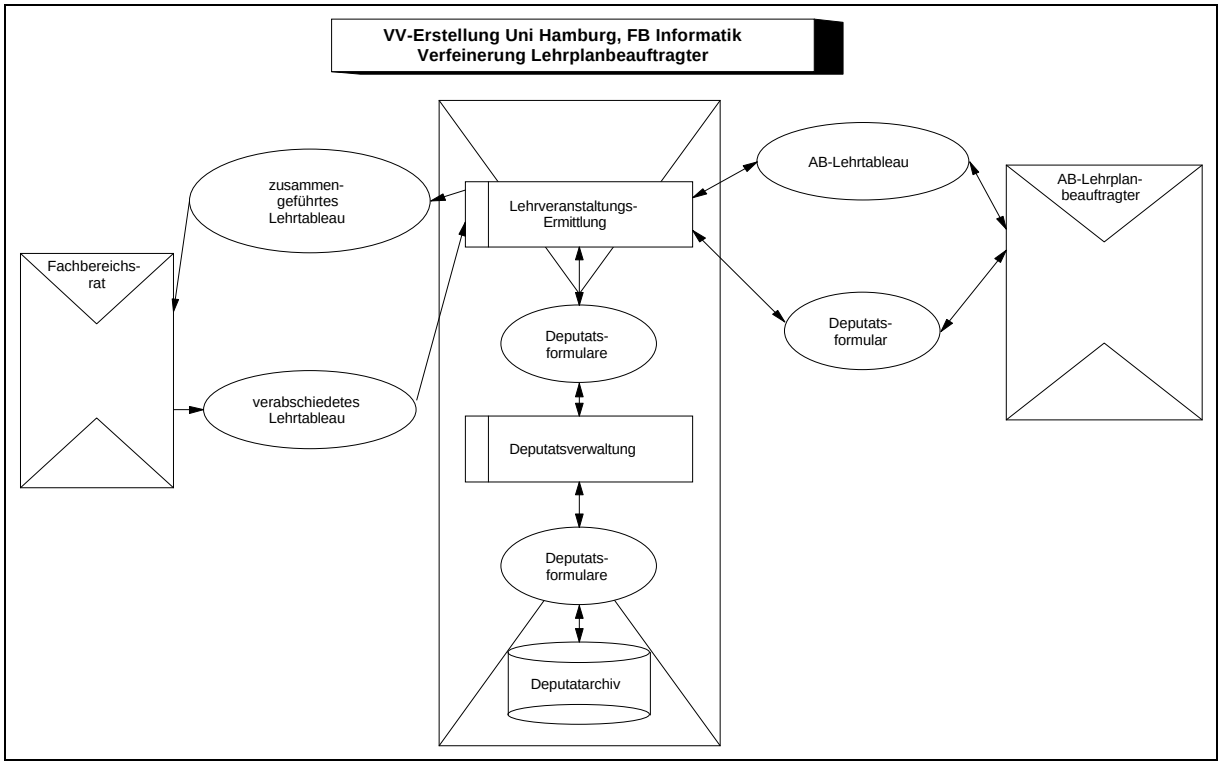

Abbildung 3.1-2: Lehrplanbeauftragter

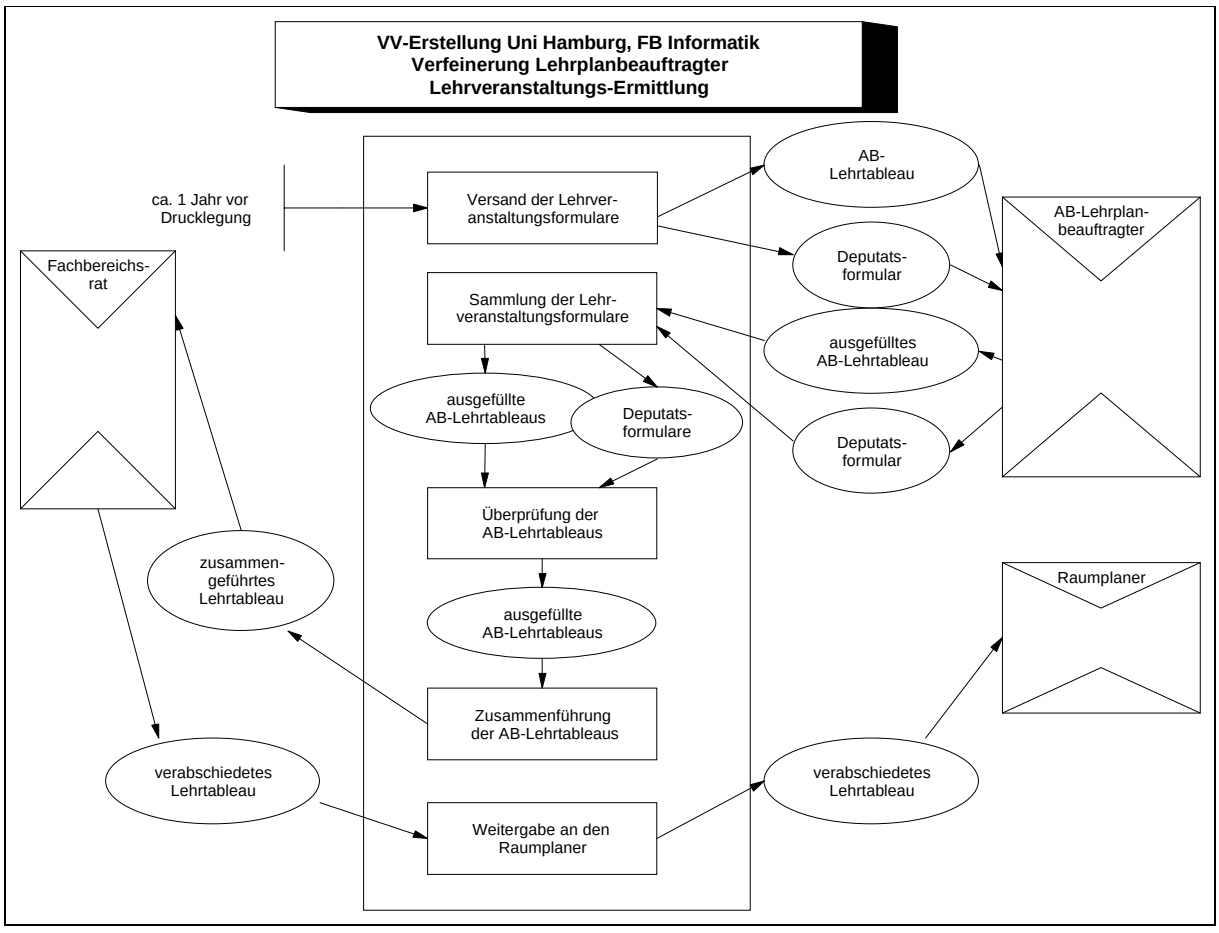

Abbildung 3.1-3: Lehrplanbeauftragter - LV Ermittlung

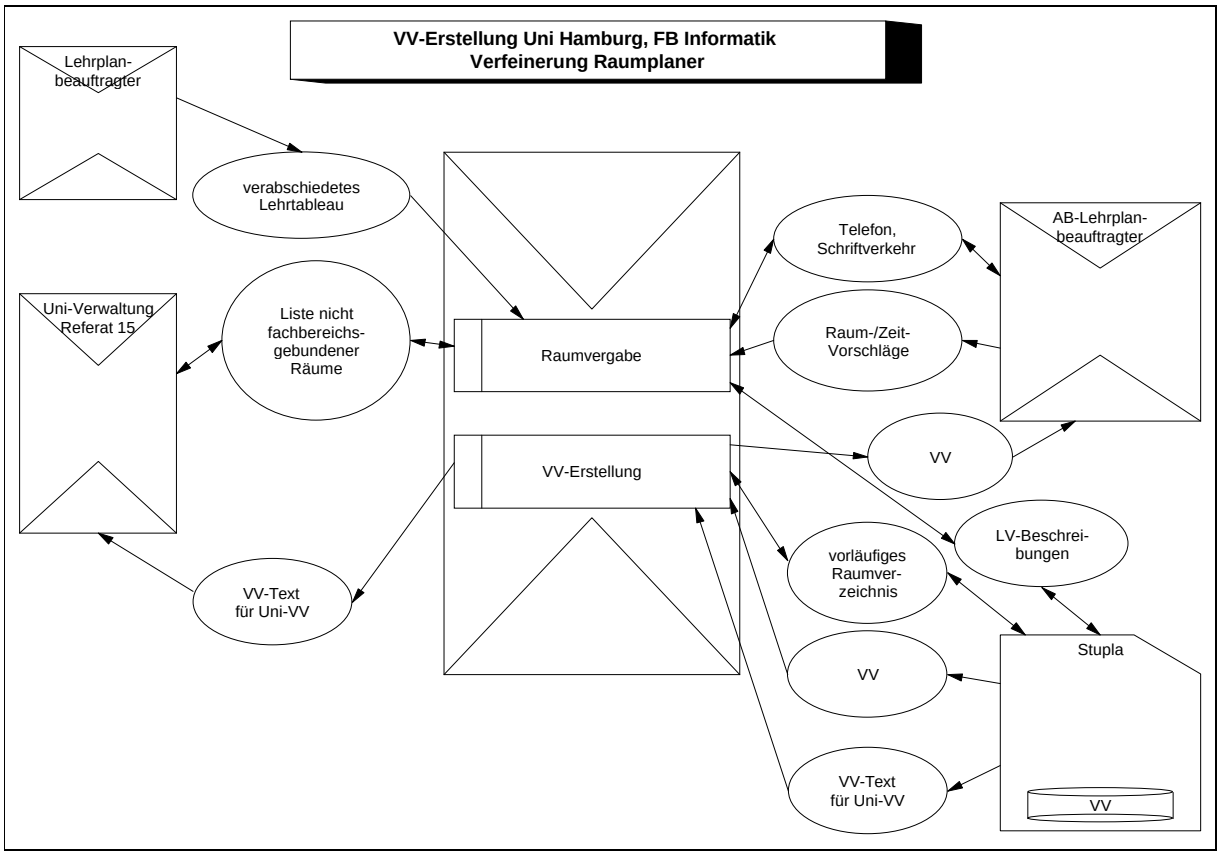

Abbildung 3.1-4: Raumplaner

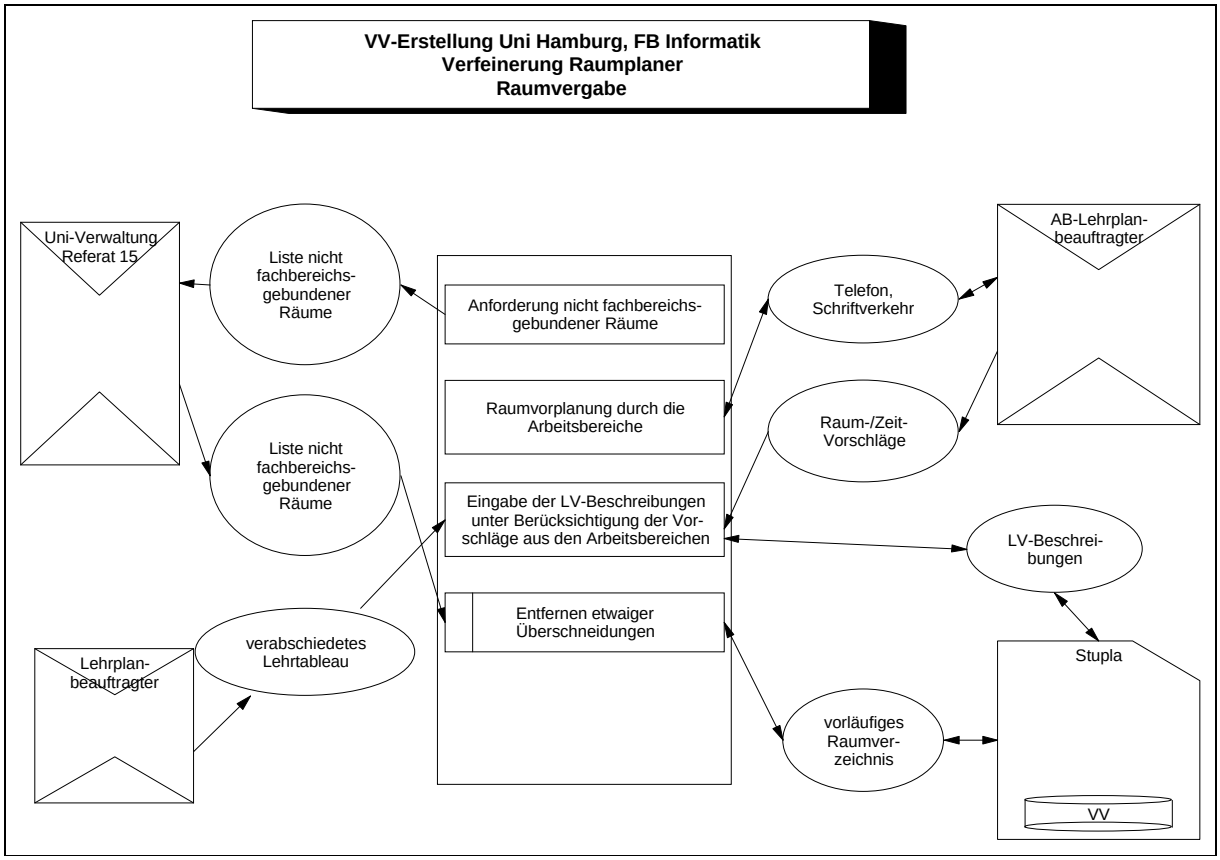

Abbildung 3.1-5: Raumplaner - Raumvergabe

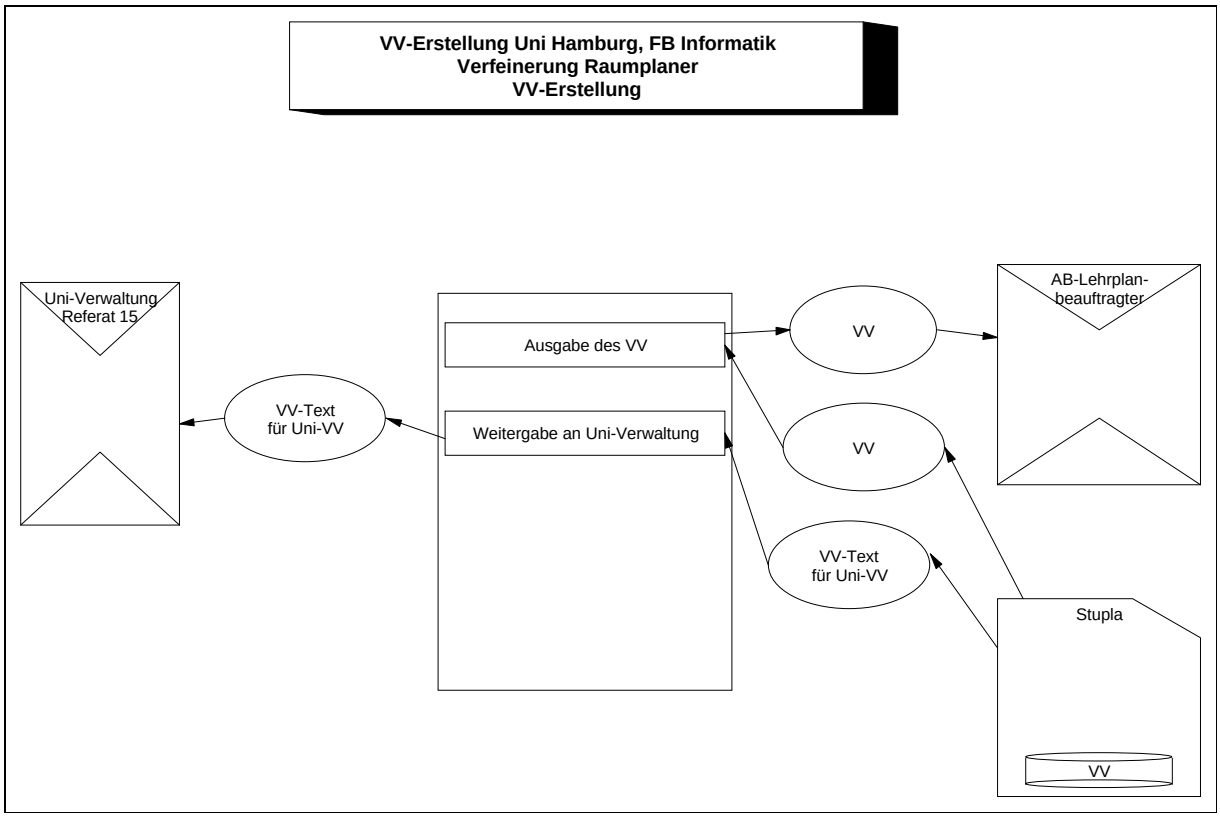

Abbildung 3.1-6: Raumplaner - VV-Erstellung

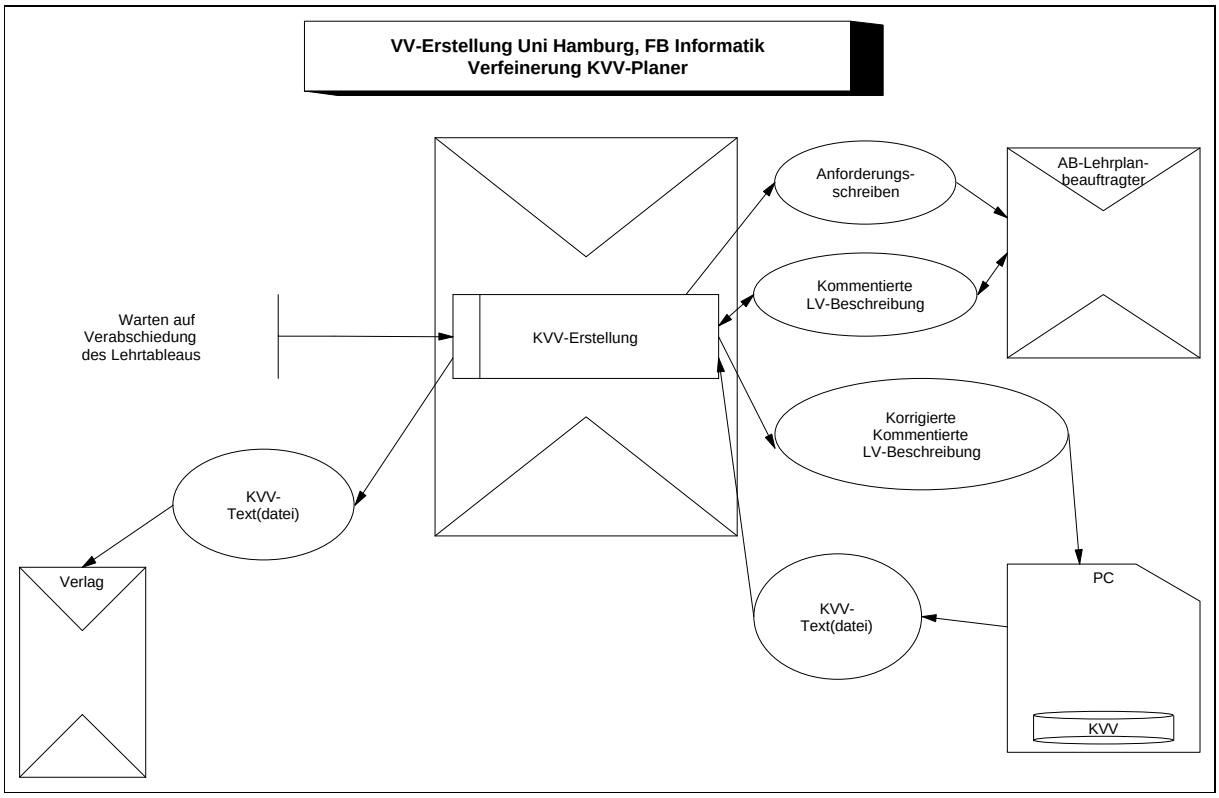

Abbildung 3.1-7: KVV-Planer

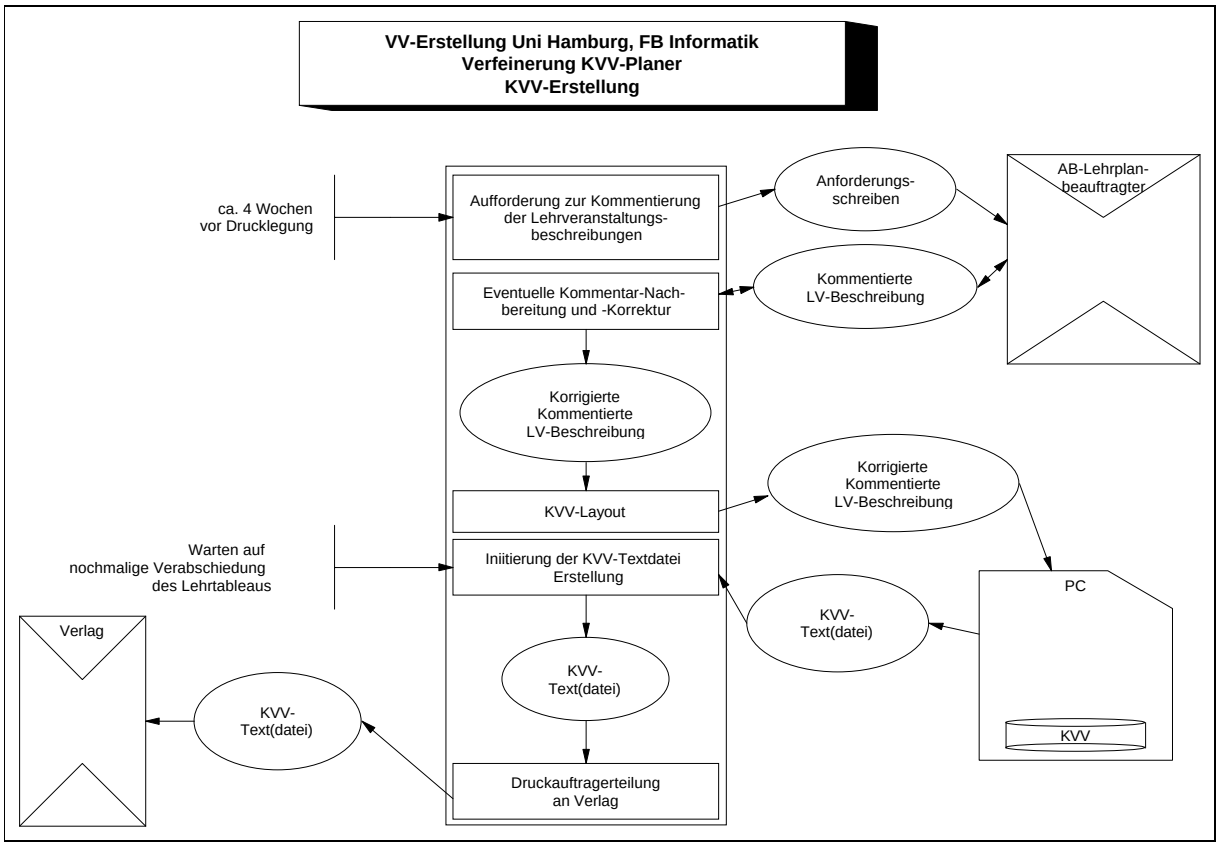

Abbildung 3.1-8: KVV-Planer - KVV-Erstellung

3.1.3 Darstellung durch Aufgabennetze: 2. Version

Die erste Version unserer Aufgabennetze haben wir zunächst in unserer Domänenmodellierungsgruppe diskutiert, um sie dann auch im Projektseminar den anderen Teilnehmern vorzustellen. Dies war von allgemeinem Interesse, da der Ablauf einer VV-Erstellung nur durch die bestehende Vorstudie in Berlin bekannt war. Erst dadurch wurde vielen bewußt, daß sie über das in Hamburg praktizierte Verfahren nicht sehr viel wußten.
Diese Diskussion war allerdings auch für uns sehr fruchtbar, da wir hier wertvolle Anregungen zu einer anderen Gestaltung der Aufgabennetze erhalten haben. Der Hauptpunkt der Kritik war vor allem:

- Im Übersichtsnetz sollten Aufgaben der funktionellen Rollen ersichtlich sein. Wir haben aus Gründen der Übersichtlichkeit in diesem Netz die Aufgaben weggelassen,[1] da wir der Meinung waren, daß zu viele Elemente in einer Grafik der Übersicht abträglich seien. Dies hatte allerdings zur Folge, daß wir bei der Vorstellung im Seminar, deren Teilnehmer ja bei der Erstellung dieser Netze größtenteils nicht beteiligt waren, in einen gewissen Erklärungsnotstand gerieten, da gerade die Darstellung der funktionellen Rollen **mit** ihren Aufgaben erst einen Überblick über das Anwendungsgebiet ermöglichen.

Dieser Autor-Kritiker-Zyklus hat uns dann dazu bewogen, die Aufgabennetze zu überarbeiten. Dabei ist uns leider ein Denkfehler unterlaufen, der an dieser Stelle kurz erläutert werden soll. Wie bereits angemerkt, haben wir angenommen, daß die zusätzliche Darstellung von Aufgaben in unseren Netzen der Übersicht über das Anwendungsgebiet eher abträglich sei. Aus diesem Grunde haben wir uns überlegt, die drei funktionellen Rollen Raumplaner, Lehrplanbeauftragter und KVV-Planer zu einer funktionellen Rolle Fachbereichsplaner (FB-Planer) zusammenzufassen und dann die Aufgaben mit in diesen Netzen darzustellen. Dieser Gedankengang wurde wohl auch noch dadurch bestärkt, daß die Berliner Gruppe ja ebenfalls ihre Aufgabennetze entwickelte und wir dadurch Einblick in ihre Arbeit hatten. Wie nämlich in Kapitel 3.2.1 ersichtlich werden wird, gibt es gerade hier keine personelle Trennung, wie sie in Hamburg üblich ist. Durch diese Überlegungen ist folgendes Übersichtsnetz entstanden.

[1] siehe Abbildung 3.1-1: Übersichtsnetz

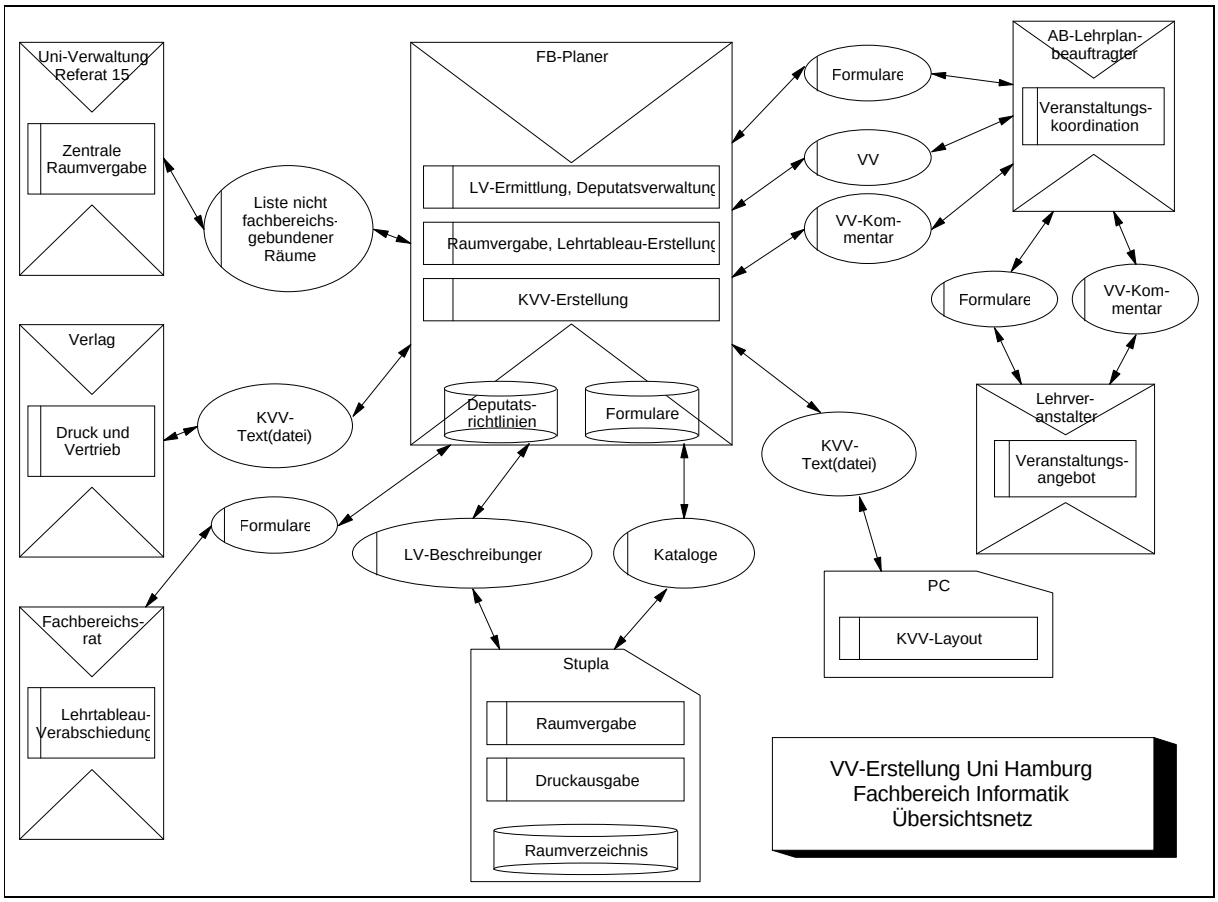

Abbildung 3.1-9: Übersichtsnetz, FB-Informatik

Weitere Verfeinerungen wollen wir an dieser Stelle nicht darstellen, da vor allem folgender Kritikpunkt gegen diese – auf den ersten Blick elegante und übersichtliche – Lösung spricht:
Es gibt konzeptionelle Probleme mit dem Verständnis des Begriffs „funktionelle Rolle“. Zwar ist es durchaus denkbar, daß eine Person mehrere funktionelle Rollen wahrnimmt. Dies ist auch in unserem Beispiel ersichtlich, da beispielsweise der Raumplaner Professor am FB Informatik und damit gleichzeitig auch Lehrveranstalter ist. Auch können mehrere Personen dieselbe funktionelle Rolle einnehmen, denn es gibt ja beispielsweise sehr viele verschiedene Lehrveranstalter. Allerdings erlaubt das Konzept der funktionellen Rolle keine Verschmelzung unterschiedlicher funktioneller Rollen. „Eine funktionelle Rolle besteht aus einer Klasse zusammengehöriger Aufgaben, für die eine Person zuständig ist.“[1]
Eine kurze Erklärung zu diesem Punkt ist auch leicht nachzuvollziehen: Die von uns erstellten Aufgabennetze sollen hier ja den Charakter eines Ist-Zustandes haben, und insofern ist es an dieser Stelle äußerst problematisch, neue Begrifflichkeiten und Umstrukturierungen vorzunehmen, die es in der Realität einfach nicht gibt.
Aus diesen Gründen sind wir also von dieser Lösung wieder abgekommen und mußten uns nun doch näher mit unserer ersten Lösung beschäftigen, obwohl es der Übersichtlichkeit abträglich erscheint. Nachfolgend soll an dieser Stelle nun noch einmal das Übersichtsnetz mit den neu eingearbeiteten Aufgaben dargestellt werden.

[1] vgl. Floyd (a.a.O.), S. 62

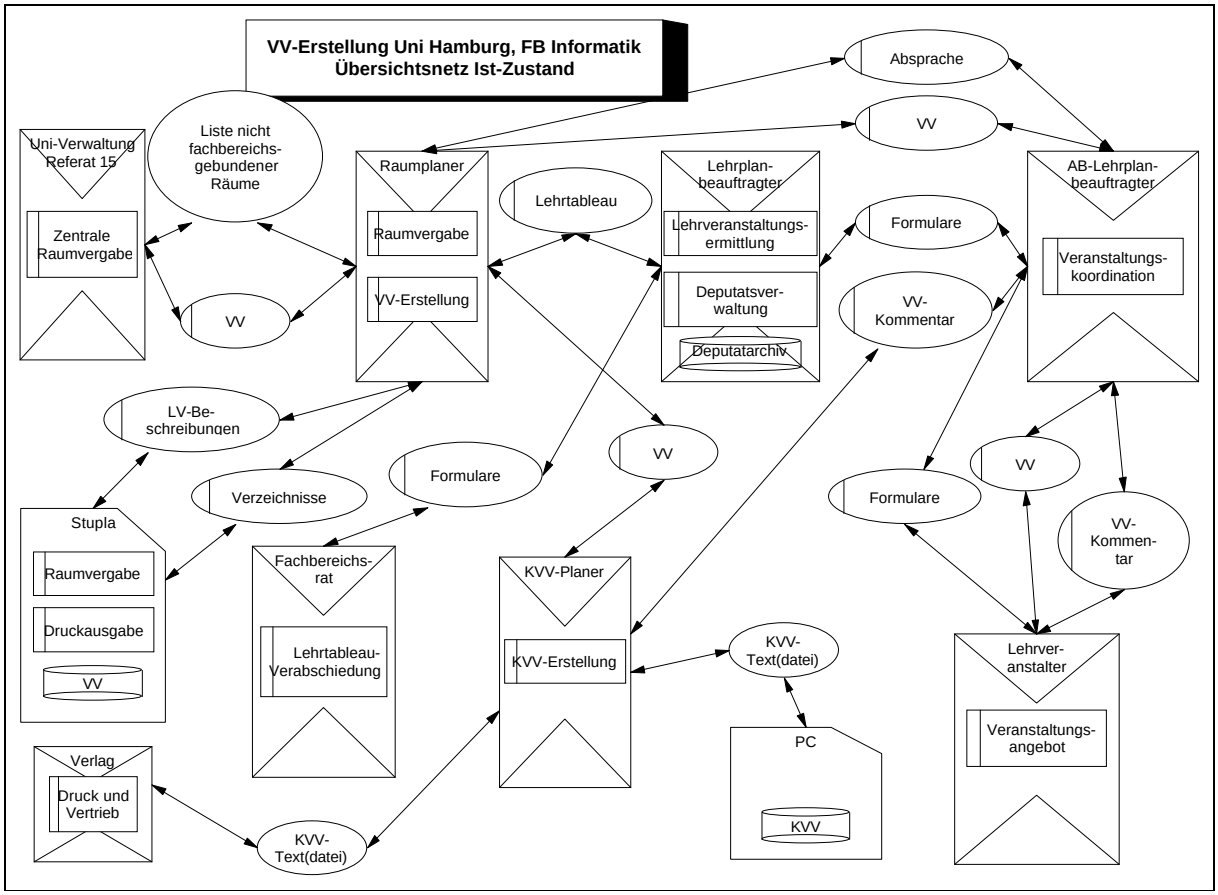

Abbildung 3.1-10: Übersichtsnetz, FB-Informatik - Endversion

3.2 Ist-Zustand Berlin

Die Informationen zur Veranschaulichung eines Ist-Zustandes für Berlin waren ungleich dürftiger als die für Hamburg zur Verfügung stehenden Quellen. Als Material ist vor allem die Vorstudie[1] zu nennen. Ergänzt wurden diese Ausführungen durch einige Aussagen der Mitarbeiter des Arbeitsbereichs Softwaretechnik. Dieser Arbeitsbereich ist nämlich 1991 von der TU Berlin zur Universität Hamburg gewechselt, und dadurch waren noch gewisse Details der VV-Erstellungsprozedur rekonstruierbar.

Die beiden Ist-Zustände in Hamburg und Berlin sollten durch die grafische Darstellung von Aufgabennetzen besser miteinander vergleichbar gemacht werden. Auf dem Weg zu solchen Netzen sind von uns zusätzlich Szenarios geschrieben worden. Diese sind vergleichbar mit den für Hamburg geschriebenen.[2] Sie dienen zu einem Großteil nun dazu, grundlegende Begrifflichkeiten und Zusammenhänge einzuführen und zu erläutern. Auch diese Szenarios haben sich natürlich der Kritik zu stellen, da sie hauptsächlich aus den eben genannten Materialien erstellt worden sind. Dadurch erhalten sie eher den Charakter einer Sekundärquelle. Aus diesem Grunde sind leider viele Angaben nicht erschöpfend und wirken ungenau. Wir waren allerdings der Meinung, daß die Informationen ausreichen, um einen guten Überblick über das Berliner Vorgehen bei der VV-Erstellung zu erlangen.

Vorweg sei gesagt, daß in Berlin schon seit Jahren ein EDV-System zur Unterstützung der VV-Erstellung eingesetzt wird. Leider konnten wir über dessen genauen Umfang und Aussehen keine weiteren Informationen gewinnen.

[1] siehe Anhang C
[2] siehe Kapitel 3.1

3.2.1 Ist-Szenarios[1]

Szenario: Anmeldung der Lehrveranstaltungen

Die *zentrale Universitätsverwaltung* (*ZUV*) verschickt an die Institute der Fachbereiche vier verschiedene Dokumente, mit deren Hilfe dem formalen Weg der Anmeldung von Lehrveranstaltungen Genüge getan wird. Dies sind im einzelnen folgende *Lehrveranstaltungsbelege* (*LV*): Der *Erfassungsbeleg* dient zur Anmeldung neuer Lehrveranstaltungen, die bisher noch nicht elektronisch erfaßt worden sind. Daneben werden *Lektionsbögen* verschickt, wobei jeder einzelne Lektionsbogen genau eine bereits gespeicherte und damit in einem bestimmten Turnus wiederkehrende Veranstaltung beschreibt. Die darauf befindlichen Daten müssen meist nicht neu erfaßt, sondern nur bestätigt oder gegebenenfalls aktualisiert werden. Als zusätzliche Unterlagen werden zwei *Schlüsselverzeichnisse* versandt, mit deren Hilfe sich *Hochschulbereiche* und *Zwischenüberschriften* ermitteln lassen. Die Zwischenüberschriften dienen dazu, durch einen 4-stelligen Schlüssel sowohl einen Fachbereich als auch eine dazugehörige Forschungseinheit mit der Unterteilung in Grund- und Hauptstudium eindeutig zu identifizieren. Die Schlüsselverzeichnisse sind also Hilfsmittel für die Erstellung von Erfassungsbögen.
Der *Stundenplanbeauftragte des Fachbereichs* (*FB-Planer*) leitet diese von der ZUV erhaltenen Unterlagen an die einzelnen Institute seines Fachbereiches weiter, die sie dann wiederum den eigentlichen Lehrveranstaltern zukommen lassen. Die Lehrveranstalter beantragen durch Ausfüllen von Erfassungsbögen neue Lehrveranstaltungen und bestätigen oder ändern ihre in den Lektionsbögen erfaßten wiederkehrenden Lehrveranstaltungen. Nicht wieder eingereichte Lektionsbögen führen im übrigen dazu, daß die entsprechend gespeicherten Daten der Lehrveranstaltung gelöscht werden. Die Erfassungsbögen müssen für ein Wintersemester bis spätestens 1. April bzw. für ein Sommersemester bis zum 1. Oktober dem entsprechenden Stundenplanbeauftragten wieder vorliegen. Dieser überprüft, ob die LV-Formulare korrekt ausgefüllt sind, und sendet sie dann an die ZUV zurück. Diese gibt dann die handschriftlich geführten Erfassungs- und Lektionsbögen an eine externe Firma weiter, die sie rechnergestützt erfaßt.

Szenario: Koordination

Die erfassten Daten kommen zur ZUV zurück und werden in den dortigen Rechner eingespielt. Die Daten werden von zwei Programmen bearbeitet. Ein Textdatenprogramm bearbeitet die Angaben, die später im VV gedruckt werden, und ein Systemdatenprogramm verwaltet die Einzelheiten, die für die Planung relevant sind. Dieses Programm übernimmt die *Raum-Zeit-Planung* und arbeitet dabei nach bestimmten Prioritäten. So werden Pflichtfächer der einzelnen Studiengänge so gelegt, daß sie sich nicht überschneiden. Diese Pflichtfächer sind durch Studienordnung und Fachbereichsbeschlüsse eindeutig identifiziert. Ergebnis dieses Vorgangs ist der *Raumbelegungsplan*. Für Wahlpflichtfächer wird versucht, diese soweit überschneidungsfrei zu halten, wie es „sinnvoll“ erscheint. Diese Frage wird im persönlichem Gespräch mit den veranstaltenden Dozenten beantwortet. Dazu werden nach und nach die entsprechenden Dozenten, für deren Veranstaltungen es Überschneidungen gibt, telefonisch kontaktiert. Ergebnis dieser Absprache ist der Ausdruck von *Korrekturfahnen* durch die ZUV, die an die entsprechenden Veranstalter geleitet werden. Sie enthalten die geänderte Raum- und/oder Zeitangabe, aber auch alle weiteren Daten. Die Dozenten können einzelne Angaben wie Beschreibungstext oder Anzahl der Teilnehmer noch korrigieren und bestätigen abschließend diese neuen Termin- und Raumvorgaben. Nachdem die Korrekturfahnen wieder bei der ZUV eingegangen sind, wird die Stundenplangestaltung überarbeitet.

[1] Die hier auftretenen Begriffe werden in einem Glossar, das sich in Anhang A und B befindet, genauer erläutert.

Szenario: Abschließende Tätigkeiten

Nach dem ersten Abschluß der Raum- und Zeitvergabe wird ein erster Computerausdruck an die entsprechenden Stundenplanbeauftragten der Fachbereiche gesandt. Diese stellen sie dann den entsprechenden Dozenten zur Verfügung. Diese Daten werden von den Lehrveranstaltern darauf hin überprüft, ob die Daten mit den eingereichten Erfassungs- bzw. Lektionsbögen übereinstimmen. Es können zu diesem Zeitpunkt durch Änderungen auf dem Computerausdruck noch Modifikationen einfließen. Sogar ganze Lehrveranstaltungen können noch angemeldet werden. Dies ist beispielsweise dann sinnvoll, wenn Erfassungsbelege vergessen worden sind. Diese Unterlagen werden von den Stundenplanbeauftragten wieder eingesammelt und müssen innerhalb einer bestimmten Frist der ZUV wieder vorliegen.
Diese korrigierten LV-Daten werden in das Programm eingegeben und bestimmen dadurch nun den endgültigen, nicht mehr änderbaren Raum-Zeit-Plan, der dem VV entspricht. Dieser endgültige Stundenplan wird als Computerausdruck wieder an die einzelnen Institute geschickt, und zusätzlich werden nun die LV-Daten mit Hilfe eines Textverarbeitungsprogramms für das kommentierte Vorlesungsverzeichnis aufbereitet.

3.3 Darstellung durch Aufgabennetze

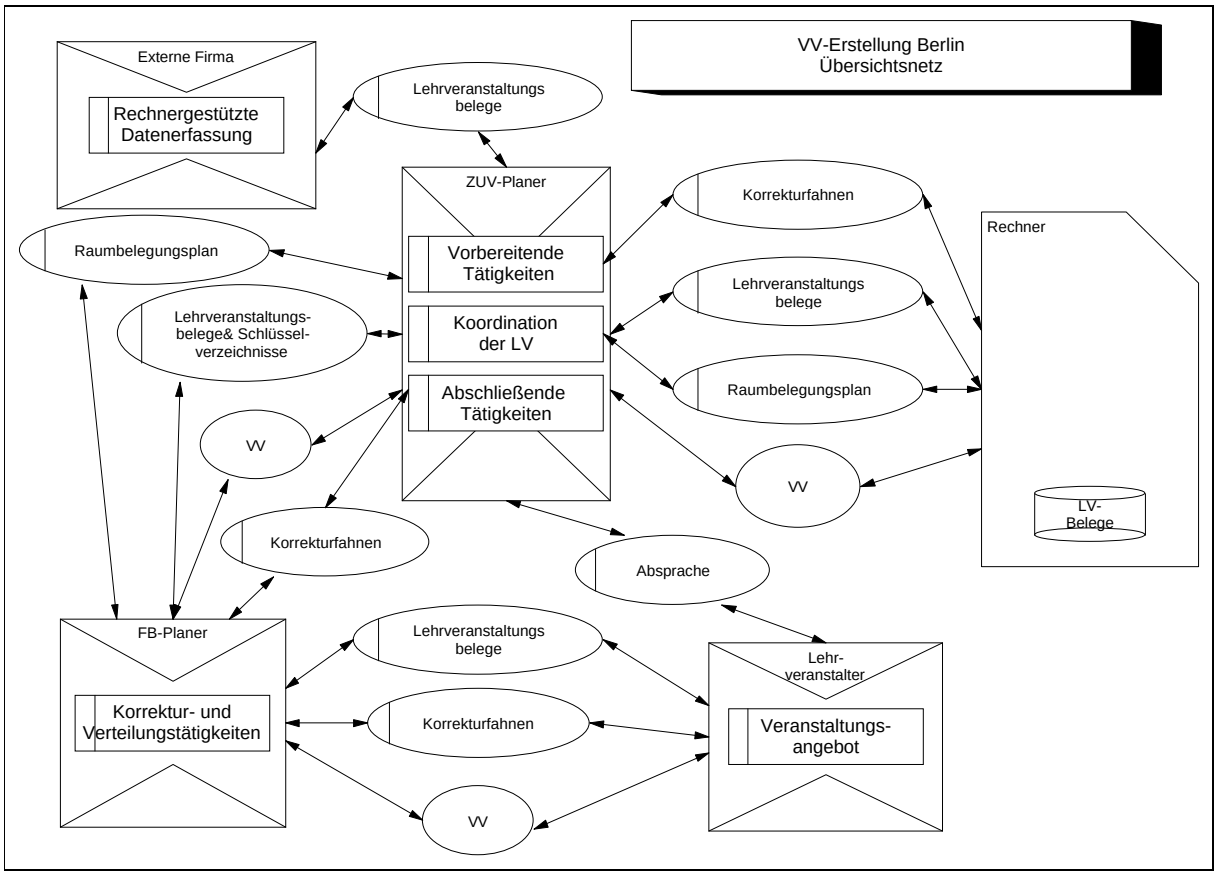

Abbildung 3.3-1: Übersichtsnetz Berlin

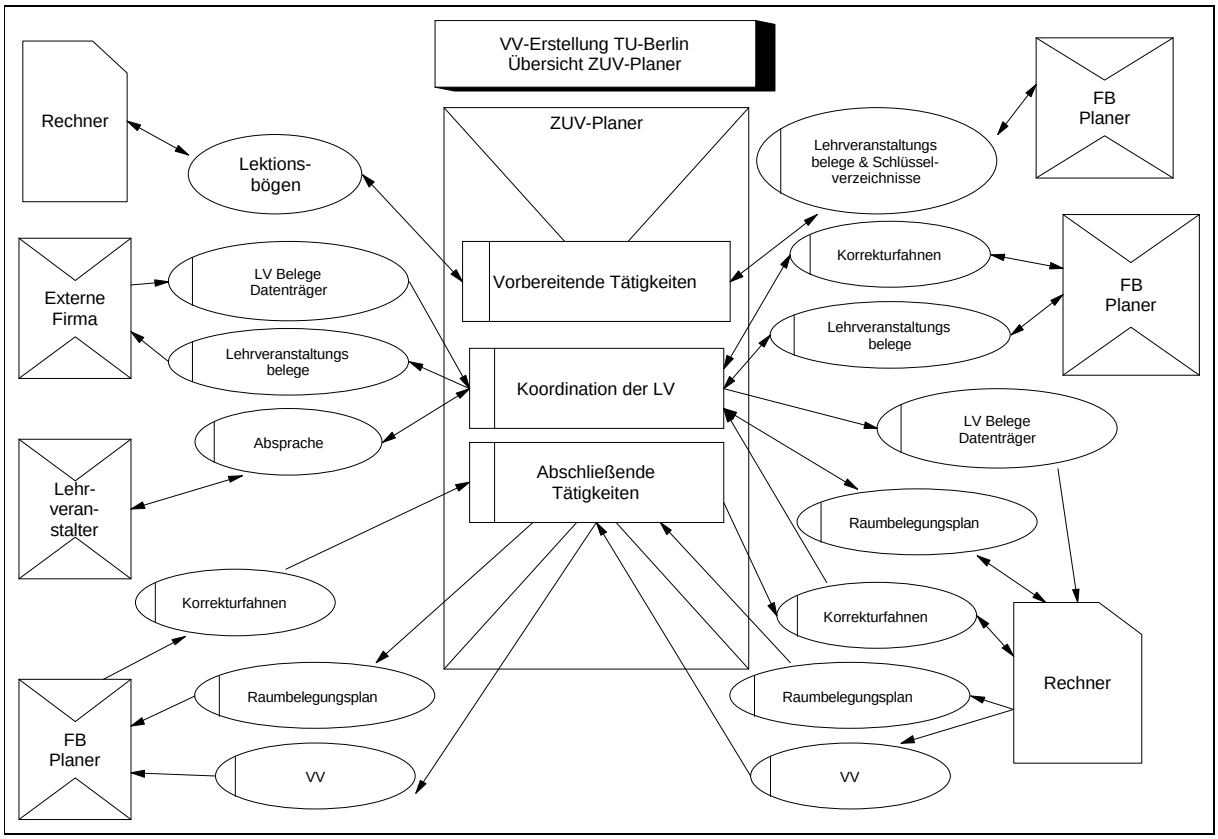

Abbildung 3.3-2: Übersicht ZUV-Planer

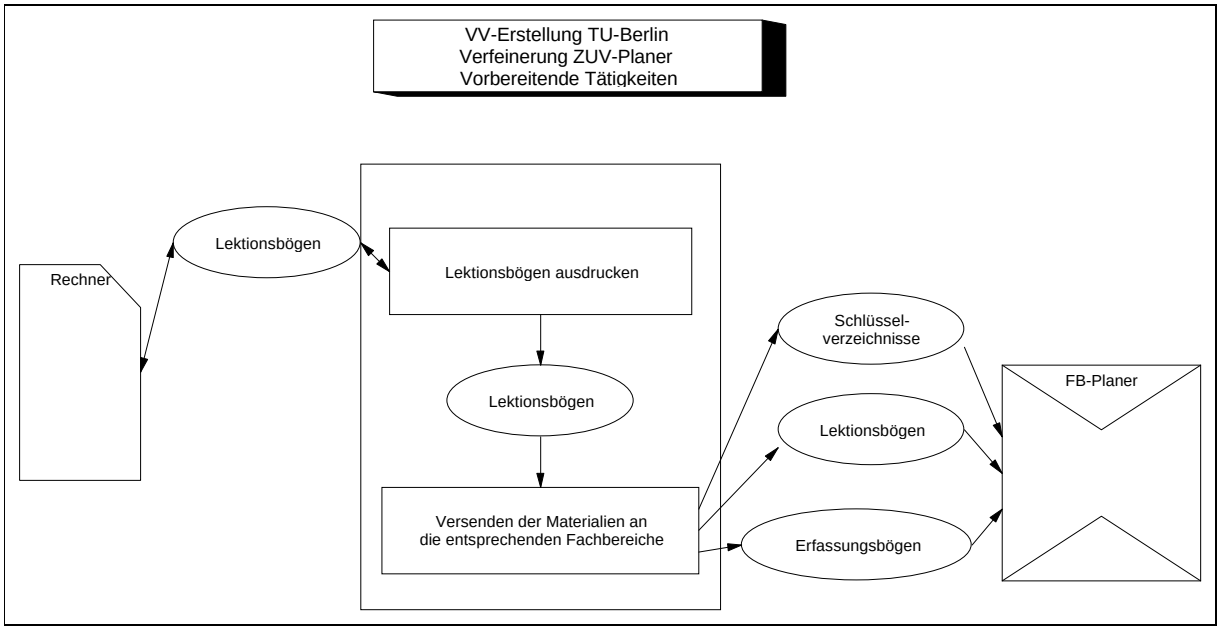

Abbildung 3.3-3: ZUV-Planer - Vorbereitende Tätigkeiten

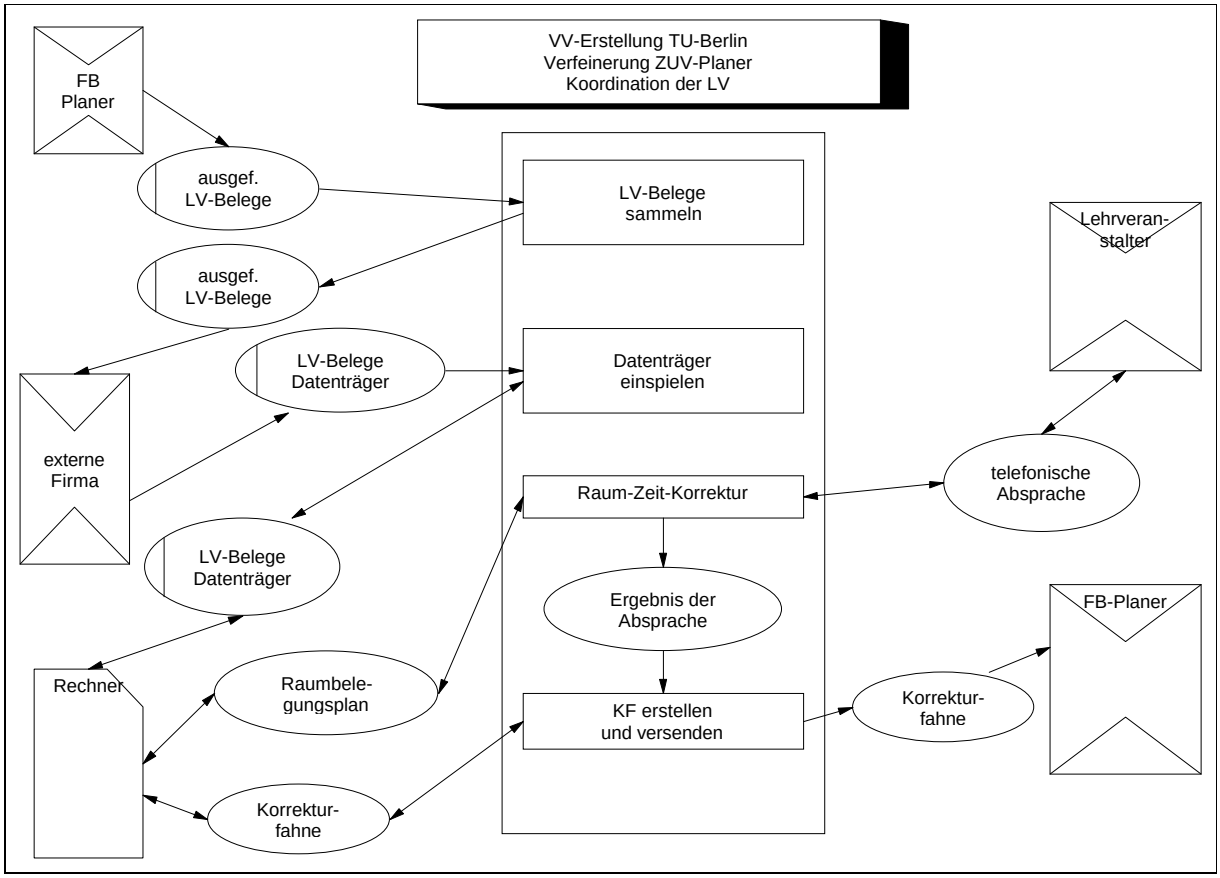

Abbildung 3.3-4: ZUV-Planer - Koordination

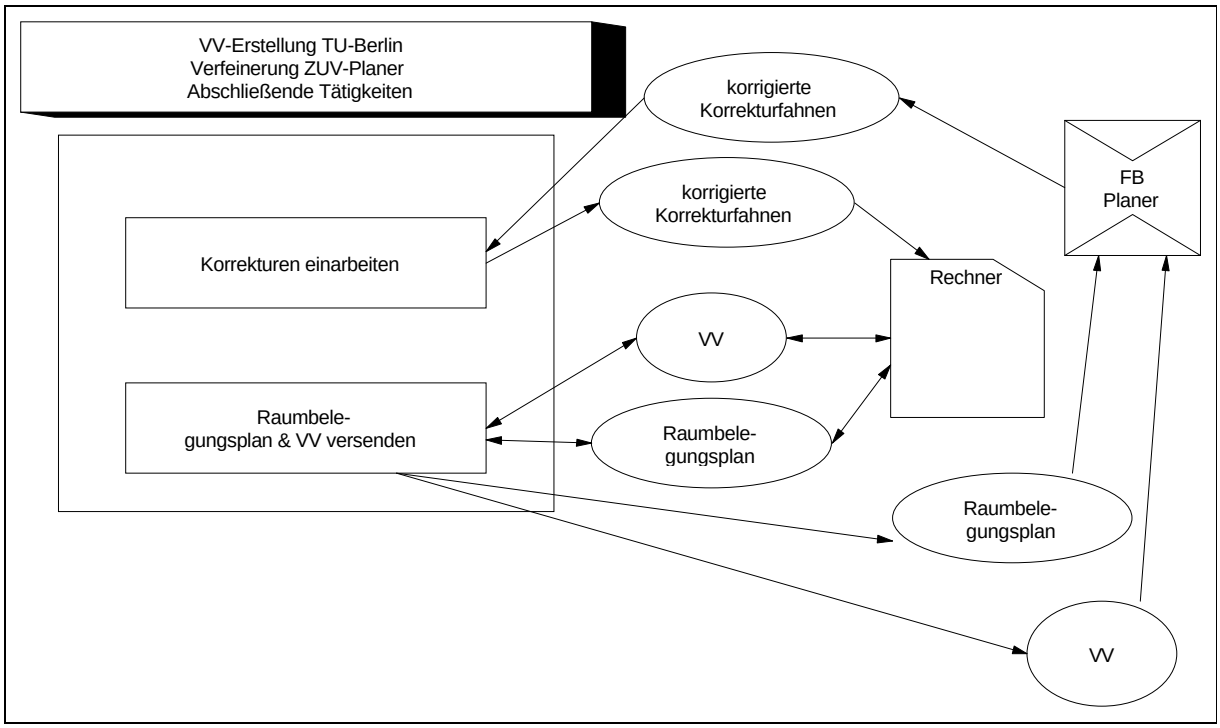

Abbildung 3.3-5: ZUV-Planer - Abschließende Tätigkeiten

4 Vergleich zwischen Hamburger und Berliner Modell

Im letzten Abschnitt haben wir Ist-Zustände der VV-Erstellungen in Berlin und Hamburg dargelegt. Dieses Kapitel hatten wir bei der folgenden Untersuchung immer im Hinterkopf, da es als wichtige Grundlage zur Bildung unseres Domänenmodells und aller weiteren Domänenanalyse-Aktivitäten[1] zu sehen ist. Die Analyse dieser beiden Beispiele soll nun in diesem Abschnitt beschrieben werden, wobei wir zunächst klären wollen, ob und wie die Aufgabennetze zusammen mit den Szenarios des letzten Abschnittes helfen können, unsere bereits vorgestellten Aktivitäten zu unterstützen.
Uns war beim Erstellen der Aufgabennetze in Kapitel 3 klar, daß wir damit versuchen, einige Aktivitäten unseres DA-Ansatzes besser in den Griff zu bekommen. Explizit waren dies hauptsächlich die **Auswahl repräsentativer Beispiele aus der Problemdomäne** und das **Erlernen der Fachsprache/Fachwissen**. Hierzu ist anzumerken, daß die Auswahl der Beispiele durch Art und Umfang des Projektseminars vorgegeben waren, und daß wir hoffen, daß unsere Leser aus dem letzten Kapitel wenigstens einen groben Überblick über die praktizierten Vorgehensweisen mit in diesen Abschnitt retten konnten.
Es sei auch hier nochmals explizit darauf hingewiesen, daß die folgenden Aktivitäten nicht als einzelne aufeinanderfolgende Schritte zu sehen sind. Sie sind vielmehr als Komponenten zu begreifen, die ineinander verzahnt sind und sich daher teilweise gegenseitig stark beeinflussen. Dieses zyklische Vorgehen ist schriftlich schwer zu beschreiben, daher werden die Methoden und Ergebnisse hier thematisch zusammengefasst.

4.1 Analyse der Beispiele

Der erste Schritt bei der Analyse der Beispiele hat vor allem zum Ziel, Unterschiede und Gemeinsamkeiten der verschiedenen VV-Erstellungen herauszuarbeiten. Die Aufgabennetze haben uns dazu als Diskussionsgrundlage gedient. Sie halfen unserer Gruppe, verschiedene Standpunkte, was Ähnlichkeiten bzw. Gegensätze angeht, den anderen Gruppenmitgliedern gegenüber und auch sich selber beispielhaft darzulegen. Ergebnis dieser Diskussion sind die beiden nachfolgenden Ausführungen über Unterschiede und Gemeinsamkeiten.

Unterschiede in der VV-Erstellung

Universität Hamburg, FB Informatik	Technische Universität Berlin
Unterschiedliche Aufbauorganisation:	
Drei funktionelle Rollen für die VV-Erstellung: - Raumplaner - Lehrplanbeauftrager - KVV Planer	Alleinige Planungstätigkeiten durch den ZUV-Planer.
Planung liegt im Schwerpunkt bei den einzelnen Fachbereichen. Diese Individualität geht sogar soweit, daß die einzelnen Arbeitsbereiche ihre Räumlichkeiten eigenverantwortlich verplanen. Darin liegt auch der Verzicht auf die Nutzung der automatischen Raumvergabe bei dem Stundenplanprogramm Stupla begründet.	ZUV übernimmt Planung. FB-Planer nehmen nur Korrektur- und Verteilungsfunktionen wahr. Hier wird auch die rechnergestützte Raum- und Zeitplanung durchgeführt, da in Berlin der gesamte Vorgang zentralisierter abläuft.

[1] siehe Kapitel 2.5

Universität Hamburg, FB Informatik	Technische Universität Berlin
Nur mündliche Absprache und anschließende Einarbeitung der Änderungen in das VV.	Verstärkter Rechnereinsatz: Mündliche Absprache und Ausdruck von Korrekturfahnen bei Verlegungen von Veranstaltungen.
Wiederkehrende Veranstaltungen werden nicht vom Rechner verwaltet.	Wiederkehrende Veranstaltungen werden mit Lektionsbögen verwaltet (=Rechnereinsatz).
Doppelte Verabschiedung der Lehrveranstaltungen durch den Fachbereichsrat. Zunächst wird das Lehrtableau und erst kurz vor Fertigstellung das VV für gültig erklärt.	Vorgehen der Verabschiedung ist uns durch die Vorstudie nicht deutlich geworden.
Kommentierung des VV ist ein eigener Arbeitsgang und wird erst direkt vor Drucklegung begonnen. Dafür handelt es sich um eine sehr ausführliche Beschreibung.	Kommentierung wird schon in den Erfassungs- bzw. Lektionsbögen vorgenommen und ist grundsätzlich nur als Kurzbeschreibung gedacht.
Es findet eine Überprüfung statt, ob eine bestimmte Mindeststundenzahl von jedem einzelnen Lehrveranstalter durchgeführt worden ist (Deputatsverwaltung).	Es findet keine derartige Überprüfung statt; es scheint solche Auflagen in dieser Form nicht zu geben.
Es finden sich verschiedene Gegenstände, die in der Form in Berlin nicht vorzufinden sind. Einige Beispiele: - Liste nicht fachbereichsgebundener Räume - Deputatsrichtlinien	Es finden sich verschiedene Gegenstände, die in der Form in Hamburg nicht vorzufinden sind. Einige Beispiele: - Lektionsbögen, Erfassungsbögen - Korrekturfahnen

Gemeinsamkeiten in der VV-Erstellung

Die Gewinnung von Gemeinsamkeiten ist schwieriger gewesen als die der Unterschiede, da abgeklopft werden mußte, inwieweit unterschiedliche Begriffe in Hamburg und Berlin im wesentlichen wohl dasselbe meinen. Beide Universitäten haben nämlich ihr spezifisches Fachvokabular, das sich selbst bei offensichtlichen Übereinstimmungen in wesentlichen Teilen sowohl vom Wortgebrauch, als auch vom Inhalt her unterscheidet. So verbirgt sich hinter dem VV in Hamburg ein Raum-Zeit-Plan, der aber auch für die Prüfung auf Einhaltung von Deputaten verwendet wird. In Berlin ist das VV hingegen schon kommentiert; es gibt aber keine zu beachtenden Deputatsrichtlinien. Dieser Abstand der Begriffe ist unterschiedlich groß und kann kaum bestimmt werden. Dies macht deutlich, daß in diesem Zusammenhang wirklich nur von Gemeinsamkeiten, aber nicht von Gleichheiten in Hamburg und Berlin gesprochen werden kann. Die Frage ist dann natürlich, wie man trotz einer solchen Problematik die analysierten Beispiele in ein Domänenmodell überführt. Dazu kommen wir in Abschnitt 4.3.
Zu einigen Gemeinsamkeiten sei hier angemerkt:

- Ergebnis der Planung ist in Hamburg und Berlin ein Vorlesungsverzeichnis, das Veranstaltungen nach Schlüsseln sortiert und in Papierform wiedergibt. Dies geschieht zusammen mit einer Beschreibung der Veranstaltung (Beschreibung, Dozent, Teilnehmeranzahl, etc.) und der Angabe, in welchem Raum sie zu welcher Zeit stattfindet.
- Prioritätsüberprüfungen bei Pflichtveranstaltungen finden sowohl in Hamburg als auch in Berlin statt. Dies heißt, daß bestimmte Vorlesungen nicht zur selben Zeit stattfinden dürfen.

- Bei Überschneidungen wird die Auflösung dieser Doppelbelegungen hauptsächlich durch ein persönliches Gespräch mit dem Veranstalter geklärt.

4.2 Abgrenzung der Problemdomäne

Schon während der Erstellung der Aufgabennetze und noch stärker während der Analysephase wurde von uns diskutiert, was denn nun alles in unsere Problemdomäne fallen soll. So war uns von vornherein klar, daß der eigentliche Druck und Vertrieb des VV, wie er durch einen Verlag durchgeführt wird, nicht zu unserem Verständnis der Erstellung des reinen VV gehört. Hier würden wir uns zu sehr auf dem Gebiet der Textgestaltung und des Layouts bewegen.
Die Problematik der Abgrenzung wurde dann auch immer stärker im Zusammenhang mit dem Themengebiet der Erstellung unseres Domänenmodells diskutiert.

4.3 Erstellung des Domänenmodells

Für diesen Abschnitt haben wir uns zunächst überlegt, eine Art Systemvision[1] zu schreiben, die als Ziel hat, die zukünftig näher betrachteten Elemente in unserem Domänenmodell beispielhaft darzulegen. Wir sind allerdings von diesem Vorhaben abgekommen, da wir die Gefahr als sehr groß einschätzen, daß solch eine Systemvision in unserem Kontext ein Gemenge von unterschiedlichen Begriffen aus Hamburg und Berlin würde. Dieses Gemisch kann dann niemand mehr auflösen oder gar verstehen, so daß viele Mißverständnisse vorprogrammiert wären.
Stattdessen bietet sich unserer Meinung nach eher ein *Dimensionskatalog* an, der zum Ziel hat, möglichst allgemein und unabhängig von den speziellen Beispielen die später zu leistenden Komponenten eines möglichen Endsystems, mit bestehenden Restriktionen, zu erläutern. Dieser Katalog soll dann die Grundlage unseres Domänenmodells bilden. Dieser Katalog war natürlich nicht plötzlich gegeben, sondern ist als Ergebnisses unseres Domänenbildungsprozesses zu verstehen, der detailliert in den folgenden Kapiteln beschrieben wird.
Die aufgestellten Dimensionen müssen dabei nicht nur Aufgaben sein, die bereits irgendwo im untersuchten „Domänenraum“ (hier: Hamburg und Berlin) aufgetreten sind. Ein weiterer kreativer Anteil von Domänenanalytikern kann unserer Meinung nach darin bestehen, daß man gerade durch die Analyse von verschiedenen Beispielen auf übergreifende Erkenntnisse stößt, die neue Möglichkeiten bieten. So ist es etwa denkbar, die Anwender (hier: Studenten) eines Vorlesungsverzeichnisses näher zu betrachten und ihnen die Möglichkeit zu bieten, am Rechner ihren individuellen Stundenplan zu erstellen. Dies kann mit dem dort zur Verfügung gestellten VV geschehen.
Unser Dimensionskatalog wird sich aber zunächst simpler gestalten und soll vom Leser in seinen Hauptkomponenten als modular verstanden werden. Dies bedeutet, daß im später tatsächlich realisiertem System einzelne Dimensionen nur dann verwirklicht werden, wenn diese Funktionalität überhaupt erwünscht wird. Beispielsweise würde die Dimension der Deputatsverwaltung in Hamburg, aber nicht in Berlin umgesetzt werden.

[1] vgl. Kilberth et al. (a.a.O.), S.99

Dimensionskatalog

Dimension	Beschreibung
Raum- Zeitbelegung	Mindestanforderung an das spätere System. Es muß die Zuordnung von Räumen und Zeiten zu einzelnen Veranstaltungen mindestens manuell festgelegt werden können.
VV-Erstellung	Veranstaltungen werden näher klassifiziert (Vorlesungen, Seminare, etc.) und können detailliert dokumentiert werden.
Belegungskonfliktlösung	Veranstaltungen können mit Prioritäten versehen werden, wodurch rechnergestützte Vorschläge zur Raumvergabe erst Sinn machen, da nur auf diesem Wege wenigstens grobe Überschneidungen gelöst werden können.
Raumbelegung	Zu jedem einzelnen Raum kann ein Wochenplan erstellt werden, der bei dem jeweiligen Raum als Übersicht ausgehängt wird.
Deputatsverwaltung	Zu allen Veranstaltern ist ermittelbar, ob sie ihre wie auch immer geartete Pflichtstundenanzahl erfüllt haben.

5 Domänenbildung

Nachdem wir uns mit Hilfe der Aufgabennetze einen Überblick über die Vorgänge bei der Erstellung eines Vorlesungsverzeichnisses in Hamburg und Berlin verschafft hatten, wollten wir nun versuchen, aus diesem Wissen ein objektorientiertes Domänenmodell zu konstruieren. Die Probleme, auf die wir bei der Entwicklung dieses Modells gestoßen sind, und die Lösungen und Erkenntnisse, die wir dabei gewonnen haben, sollen Inhalt dieses Kapitels sein. Dabei wurde ein Teil der Ergebnisse bereits in Kapitel 4 vorweggenommen: Den dort aufgeführten Dimensionskatalog haben wir im Verlauf des im folgenden beschriebenen Domänenbildungsprozesses entwickelt.
Wie bereits ausgeführt, mußten wir feststellen, daß sich die Erstellung eines Vorlesungsverzeichnisses in Hamburg von der in Berlin z.T. gravierend unterschied. Trotzdem wollten wir versuchen, mögliche Gemeinsamkeiten herauszuarbeiten, allgemeingültige Regeln aufzustellen und fachliche Objekte zu identifizieren, die dann zu einem objektorientierten Domänenmodell zusammengefaßt werden können. Dazu muß zunächst der zentrale Begriff der Objektorientierung näher erläutert werden.

5.1 Einführung in die Objektorientierung

Während uns der Begriff der Domäne und die Methoden zur Konstruktion von Domänenmodellen zumindest theoretisch geläufig waren, beschränkte sich unser Wissen über Objektorientierung zum damaligen Zeitpunkt (während des Projektseminars) auf die Ausführungen von Bertrand Meyer in seinem Buch „Objektorientierte Softwareentwicklung“.[1] Dort bezeichnet er ein wohlorganisiertes Softwaresystem als ein „operationales Modell eines bestimmten Aspektes der Welt“ und den Softwareentwurf als „operationales Modellieren“.[2] Da der zu modellierende Aspekt der Welt aus Objekten besteht, liegt es nahe, den Entwurf auf eben diesen Objekten zu gründen: „In der physikalischen oder abstrakten Wirklichkeit sind die Objekte modelliert und warten darauf, aufgelesen zu werden!“.[3] Dieser Beschreibung Meyers nach scheint das Aufspüren von Objekten kein Problem zu sein. Beim Betrachten der Aufgabennetze waren die Objekte der Vorlesungsverzeichniswelt jedoch nicht so einfach zu entdecken. Dies lag z.T. auch an der Darstellung unserer Vorlesungsverzeichniswelt durch Aufgabennetze. Die zentrale Bedeutung der funktionellen Rollen, zwischen denen in verschiedenen Verfeinerungsstufen die zu bearbeitenden Objekte angeordnet sind, erleichtert sicherlich den Softwareentwurf nach funktionalen Ansätzen – da die Objekte jedoch in der Regel mehrfach aufgeführt sind und die Beziehungen von Objekten untereinander nicht dargestellt werden, sind die Aufgabennetze für objektorientierte Herangehensweisen eher ungeeignet.
Nach mühsamer Suche gelang es uns dennoch, erste reale Objekte zu finden, die in unser Domänenmodell zu gehören schienen. Zu diesen Objekten deklarierten wir entsprechende *Klassen*, wobei eine Klasse nach Meyer ein Sprachkonstrukt ist, das die Aspekte Modul und Typ vereinigt.[4]
Diese Identität von Modul und Typ ist eine der Eigenschaften, die objektorientierte Programmiersprachen ausmacht und sie von Sprachen wie Modula-2 unterscheidet, in denen ein Modul nur eine syntaktische Trennung von logisch zusammengehörenden Programmelementen darstellt.[5] Als weitere Eigenschaft kann eine Klasse als Einschränkung oder Erweiterung einer

[1] vgl. Meyer (a.a.O.), Kap. 4
[2] vgl. Meyer (a.a.O.), S. 55
[3] vgl. Meyer (a.a.O.), S. 55
[4] vgl. Meyer (a.a.O.), S. 67
[5] vgl. Meyer (a.a.O.), S. 66

anderen Klasse definiert werden (*Vererbungsbeziehung*), oder sie kann Objekte anderer Klassen benutzen (*Benutztbeziehung*).
Um eine möglichst klare Visualisierung der Klassen und ihrer Vererbungs- und Benutztbeziehungen zu erreichen, bedienten wir uns der von Wolfgang Strunk vorgeschlagenen *Objektdiagramme* und der selbstentwickelten *Klassenkarten*, die im folgenden vorgestellt werden sollen.

5.2 Objektdiagramme

Objektdiagramme stellen Klassen und ihre Vererbungs- und Benutztbeziehungen grafisch dar. Klassen werden dabei durch Kreise repräsentiert, die als Inschrift den Klassennamen erhalten. In der ursprünglichen Notation stehen ungerichtete Kanten für eine Benutztbeziehung, gerichtete Kanten zeigen eine Vererbungsbeziehung an. Dabei zeigt die Pfeilspitze in Richtung der Elternklasse(n).

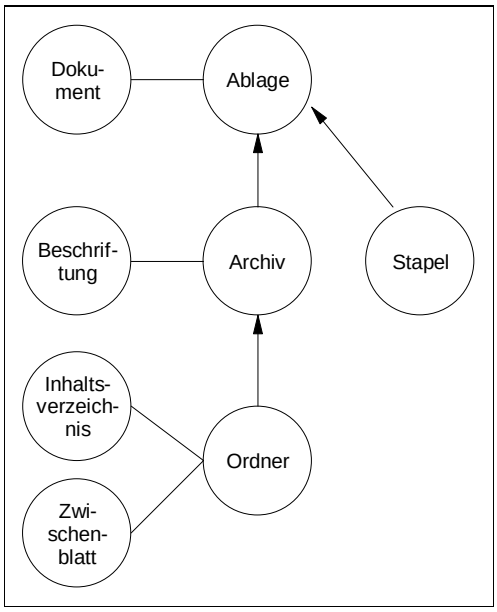

Abbildung 5.2-1: Beispiel für ein Objektdiagramm (ursprüngliche Notation)

Das Objektdiagramm in Abbildung 5.2-1 zeigt die Klassifikation von Materialien zur Ablage von Schriftstücken, wie sie typischerweise in Büroanwendungen zu finden sind.[1]
Die Klasse Ablage bildet die Wurzelklasse dieser Hierarchie und hat somit den höchsten Abstraktionsgrad. Ihre Aufgabe ist das Verwalten von Objekten der Klasse Dokument. Diese Dokumente können in die Ablage eingefügt, ausgewählt und aus der Ablage entnommen werden.
Die Klassen Archiv und Stapel erben alle Eigenschaften der Klasse Ablage.
Die Klasse Archiv benutzt ein Objekt der Klasse Beschriftung, um Archive mit einem Namen versehen zu können.
Die Klasse Ordner ist eine Spezialisierung der Klasse Archiv. Sie benutzt – zusätzlich zu den Eigenschaften von Archiv – Objekte der Klassen Inhaltsverzeichnis und Zwischenblatt.
An diesem Beispiel zeigt sich auch schon ein Problem der Objektdiagramme: Da die Benutztbeziehungs-Kanten ungerichtet sind, ist aus dem Diagramm nicht ersichtlich, welche Klasse die benutzte und welche die benutzende ist oder ob gar eine gegenseitige Benutztbeziehung vorliegt. Meist läßt die Beschriftung der Klassen nur eine dieser Beziehungen als Deutung zu, aber es können durchaus Mehrdeutigkeiten auftreten. Die am Arbeitsbereich Softwaretechnik der Universität Hamburg z.B. im Rahmen des Softwaretechnik-Projekts 94/95 verwendete

[1] vgl. Heinz Züllighoven: Arbeitsunterlagen zur Lehrveranstaltung „Objektorientierte Systementwicklung, Teil A: Entwurf“, Universität Hamburg 1994

Notation benutzt die gerichteten Kanten für die Benutztbeziehungen. Dabei zeigen die Pfeilspitzen in Richtung der benutzten Klassen. Mit den ungerichteten Kanten und einer hierarchischen Positionierung (Elternklassen stehen immer oberhalb ihrer abgeleiteten Klassen) können die Vererbungsbeziehungen eindeutig dargestellt werden. Wir haben uns deshalb dazu entschlossen, diese Notation für Objektdiagramme als Darstellungsmittel im Rahmen dieser Studienarbeit zu verwenden, zum einen aus Gründen der Eindeutigkeit, zum anderen aus der Tatsache heraus, daß es den mit der am Arbeitsbereich Softwaretechnik verwendeten Notation vertrauten Personen leichter fällt, die Diagramme zu lesen. Abbildung 5.2-2 zeigt das Objektdiagramm aus Abbildung 5.2-1 in der neuen Notation.

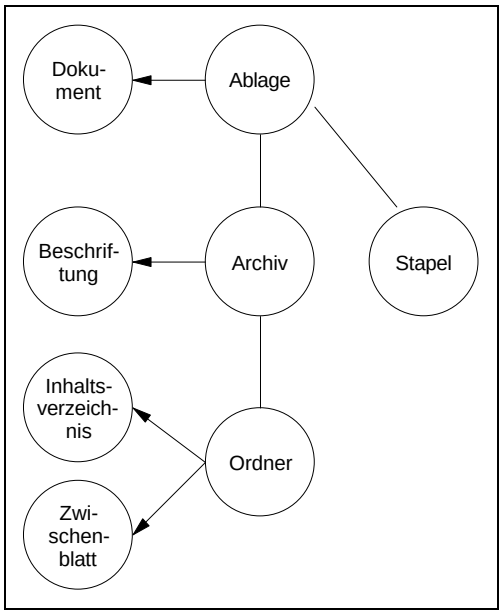

Abbildung 5.2-2: Objektdiagramm aus Abbildung 5.2-1 in der neuen Notation

5.3 Klassenkarten

Klassenkarten sollen ebenfalls die Vererbungs- und Benutztbeziehungen der Klassen darstellen, allerdings in textueller Form. Zusätzlich enthalten sie Attribute und Methoden der Klassen, wobei wir uns auf die Darstellung fachlich motivierter Methoden und Materialien beschränkt haben. Wir entwickelten dieses Darstellungsmittel in Anlehnung an die CRC-Karten von Beck und Cunningham.[1] Die CRC-Karten werden z.B. von Gamma[2] zur Unterstützung des objektorientierten Entwurfs eingesetzt.

[1] vgl. Kent Beck, Ward Cunningham: A laboratory for object-oriented thinking, SIGPLAN notices vol.24 no.10, 1989

[2] vgl. Erich Gamma: Objektorientierte Software-Entwicklung am Beispiel von ET++, Berlin u.a. 1992, Kap. 5.1.1

Abbildung 5.3-1: Klassenkarten der Klassen Ablage, Archiv, Ordner und Beschriftung

In Abbildung 5.3-1 wird ein Ausschnitt des Objektdiagramms aus Abbildung 5.2-2 durch Klassenkarten beschrieben.
Die Klasse Ablage führt eine Liste von Objekten der Klasse Dokument – eben jene Dokumente, die in dieser Ablage abgelegt sind. Es stehen Methoden zum Einfügen von Dokumenten in die Ablage, Auswählen von Dokumenten in der Ablage und Entnehmen von Dokumenten aus der Ablage zur Verfügung.
Alle Eigenschaften (Attribute und Methoden) der Klasse Ablage werden an die Klasse Archiv vererbt. Diese benutzt zusätzlich ein Objekt der Klasse Beschriftung, das über eine entsprechende Methode Beschriften() verändert werden kann.
Die Klasse Ordner wiederum ist gegenüber ihrer Elternklasse Archiv um zwei Attribute erweitert: Zum einen benutzt sie ein Objekt der Klasse Inhaltsverzeichnis, zum anderen führt sie eine Liste von Zwischenblatt-Objekten, die über die Methoden ZwischenblattEinfügen() und ZwischenblattEntnehmen() eingefügt bzw. entnommen werden können.
Die Klasse Beschriftung enthält ein Attribut Text, das über die Methoden NimmText() und GibText() gesetzt bzw. ausgelesen werden kann.

Mit Objektdiagrammen und Klassenkarten besitzen wir nun die geeigneten Darstellungsmittel für ein objektorientiertes Domänenmodell. Die Entwicklung dieses Modells geschah evolutionär in mehreren Schritten mit integrierten Autor-Kritiker-Zyklen.[1] Diese Entwicklung soll im folgenden durch eine ausführliche Beschreibung der einzelnen Entwurfsstadien und Modellentwürfe nachgezeichnet werden.

[1] vgl. Christiane Floyd et. al.: Anleitung zur konstruktiven Kritik, Einführung in die Softwaretechnik SS 1993, Universität Hamburg 1993

5.4 Erster Entwurf: Klassenidentifikation

Wie bereits angedeutet, versuchten wir zunächst, in den Aufgabennetzen Begriffe zu identifizieren, die Kandidaten für Objekte bzw. entsprechende Klassen darstellten. Als zentrale Klasse erkannten wir das Vorlesungsverzeichnis. Damit zusammen hängen Begriffe wie Lehrveranstaltung, Raumbelegungsplan und VV-Erstellung, die wir als eigene Klassen modellierten:

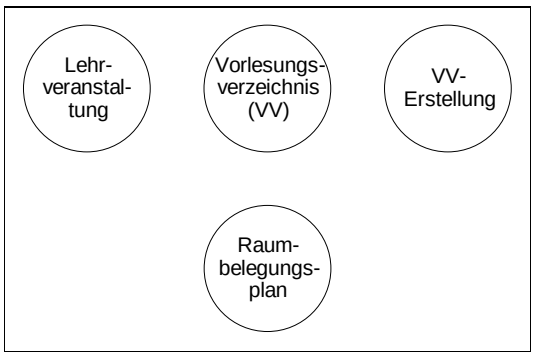

Abbildung 5.4-1: Modellierung erster Klassenkandidaten

Wird eine Lehrveranstaltung wie z.B. ein Oberseminar von mehr als einem Dozenten geleitet, so kann es durchaus vorkommen, daß nur einer der Dozenten diese Lehrveranstaltung auf sein Deputat angerechnet bekommt.[1] Für die übrigen Dozenten hat diese Veranstaltung somit keinen Lehrveranstaltungscharakter mehr. So entstand die allgemeinere Klasse Veranstaltung. Diese benutzt zunächst eine Liste von Dozent-Objekten und einen Kommentar, der die Veranstaltung näher beschreibt, jedoch kein Deputat:

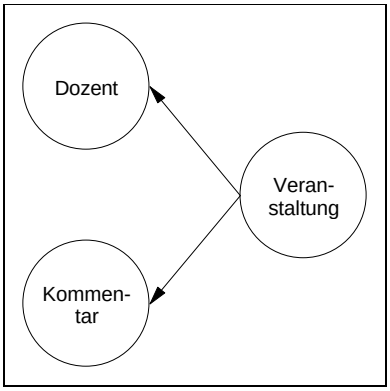

Abbildung 5.4-2: Modellierung der Klasse Veranstaltung

Die Klasse Lehrveranstaltung erbt nun von der Klasse Veranstaltung und benutzt zusätzlich ein Deputat:

[1] Es ist am Fachbereich Informatik der Universität Hamburg auch üblich, daß das Deputat auf die Dozenten aufgeteilt wird.

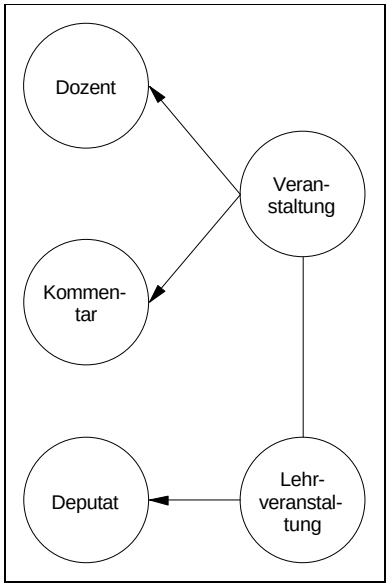

Abbildung 5.4-3: Modellierung der Klasse Lehrveranstaltung

Ein Vorlesungsverzeichnis besteht aus mehreren Veranstaltung-Objekten und ist wiederum (als Ergebnis) ein Objekt der Klasse VV-Erstellung, ebenso wie der Raumbelegungsplan:

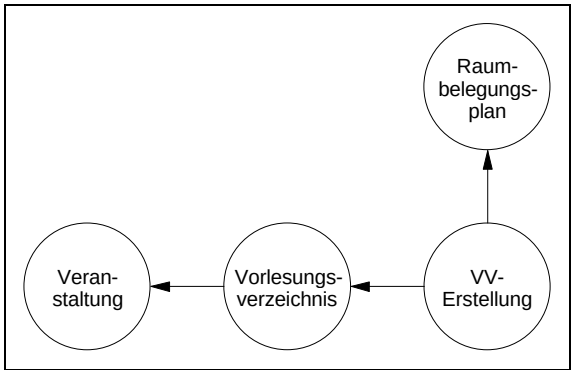

Abbildung 5.4-4: Einordnung der Klasse VV-Erstellung

Der Raumbelegungsplan besteht aus mehreren Raum-Objekten. Diese führen eine Liste von Zeitraum-Objekten, welche die Zeiträume angeben, zu denen Veranstaltungen in einem Raum stattfinden können:

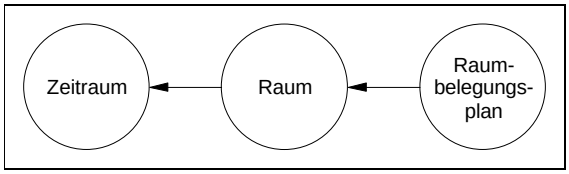

Abbildung 5.4-5: Modellierung der Klasse Raumbelegungsplan

Nun stellte sich uns folgendes Problem: Die Tatsache, daß Veranstaltungen zu bestimmten Zeiten in bestimmten Räumen stattfinden, war mit unseren bisherigen Klassen nicht zu modellieren, da ein Objekt der Klasse Raum seinen gesamten Zeitplan in Form von Zeitraum-Objekten enthält. Was wir jedoch brauchten, war eine Beziehung Raumnummer–Zeitraum. Also schufen wir eine neue Klasse Raumzeit, die ein Objekt der Klasse Zeitraum benutzt und als weiteres Attribut eine Raumnummer besitzt. Die Klasse Veranstaltung benutzt nun diese Klasse Raumzeit.

Das Zusammenfügen dieser Teilmodelle ergab einen ersten Entwurf des objektorientierten Domänenmodells, wie er in Abbildung 5.4-6 dargestellt ist.

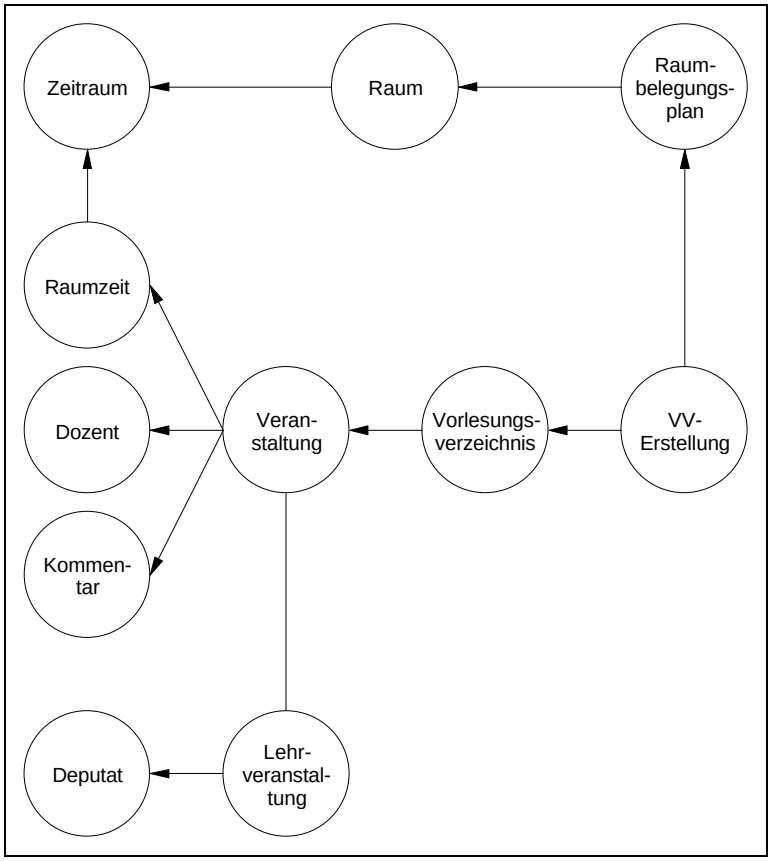

Abbildung 5.4-6: Erster Entwurf des objektorientierten Domänenmodells

Das wohl unglücklichste Konstrukt in diesem Entwurf ist die Raumzeit-Klasse, die – wie oben erwähnt – nicht in unserem Anwendungsgebiet als Klassenkandidat identifiziert wurde, sondern eine erste Lösung eines technischen Entwurfsproblems darstellt. Des weiteren stellte sich uns die Frage, wie die Klasse VV-Erstellung in einer Implementierung auszusehen hätte. Diese Klasse wurde aus den Aufgabennetzen extrahiert. Deren Fokus liegt auf funktionalen Beziehungen, so daß die Klasse weniger ein (reales) Objekt als vielmehr eine Funktion beschreibt.

5.5 Zweiter Entwurf: Autor-Kritiker-Zyklus

Das Studium des Szenarios lieferte noch weitere solcher „funktionalen Klassen“: Die Raumbelegung, das Prüfen von Lehrveranstaltungen auf Überschneidungsfreiheit, und die Deputatsverwaltung. In einem zweiten Entwurf, wie er in Abbildung 5.5-1 dargestellt ist, wurden diese neuen Anforderungen berücksichtigt.

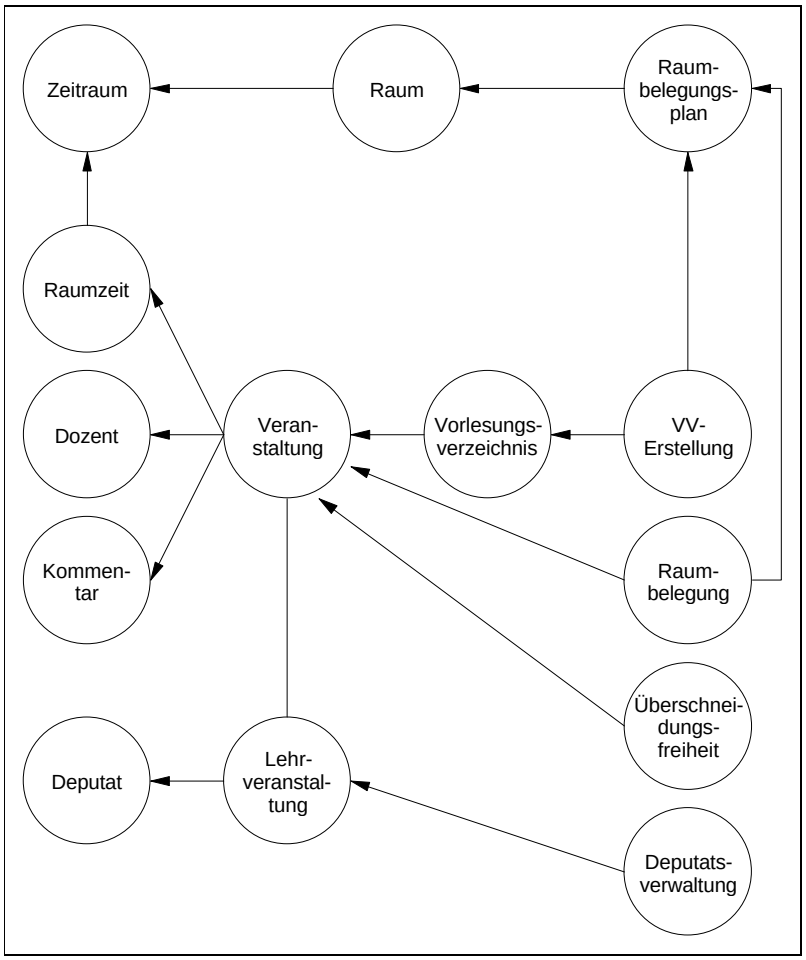

Abbildung 5.5-1: Zweiter Entwurf des objektorientierten Domänenmodells

Dieser Entwurf wurde den Seminarteilnehmern im Sinne eines Autor-Kritiker-Zyklus vorgestellt und mit ihnen diskutiert.
Auch das Plenum beurteilte die Klasse Raumzeit als unzweckmäßig. Hier mußte also eine neue Lösung gefunden werden. Die „funktionale Klasse“ VV-Erstellung sollte aufgeteilt werden in die Teiltätigkeiten „VV-Erstellung“ und „Raumbelegung“. Um diese funktionalen Klassen entwickelte sich eine Diskussion, in deren Verlauf die Berechtigung solcher Klassen grundsätzlich in Frage gestellt wurde. Es kam die Vermutung auf, daß diese Klassen vielleicht eigene Domänen sind, die dann aber nicht als Klassen modelliert werden sollten. Stattdessen sollte man die zu einer dieser Domänen gehörenden Klassen durch Umrandung gruppieren.
Da die einzelnen Domänen jeweils eine Teilaufgabe kapseln, kann man in einem späteren Softwaresystem einzelne Domänen weglassen, die für die spezielle Anwendung nicht gebraucht werden, so z.B. die Deputatsverwaltung. Dazu müssen aber die Funktionalitäten strikt getrennt und die Schnittstellen zwischen den Domänen bzw. ihren Klassen exakt definiert werden. Da dies mit den Objektdiagrammen allein nicht mehr zu visualisieren war, wollten wir für den nächsten Entwurf zusätzlich Klassenkarten als Darstellungsmittel verwenden.

5.6 Dritter Entwurf: Domänenmodellierung

Bei der Auflösung der Raumzeit-Problematik wurde die Raum-Klasse als Ausgangspunkt genommen. Ein Raum sollte nun eine Liste von Zeit-Objekten besitzen, welche die Belegungszeiten des Raumes repräsentierten. Soll nun eine Veranstaltung zu einer bestimmten Zeit stattfinden, so wird ein Verweis auf das entsprechende Veranstaltung-Objekt an das entsprechende

Zeit-Objekt in der von Raum geführten Liste von Zeit-Objekten angehängt.[1] Veranstaltung-Objekte führen eine Liste von Verweisen auf Raum-Objekte, aus denen sie ihre Veranstaltungszeiten extrahieren können. Abbildung 5.6-1 zeigt diese Zusammenhänge als Objektdiagramm.

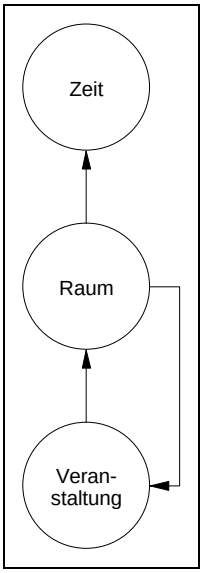

Abbildung 5.6-1: Auflösung der Klasse Raumzeit

Die gegenseitige Benutztbeziehung zwischen Raum und Veranstaltung wurde zwar als problematisch angesehen, schien aber zunächst unvermeidbar, da der Raumbelegungsplan über die Raum-Objekte auf die Veranstaltung-Objekte zugreifen mußte, während das Vorlesungsverzeichnis den umgekehrten Weg (von Veranstaltung- auf Raum-Objekte) geht. Eine ähnliche Situation ergab sich bei der Klasse Dozent. Veranstaltung-Objekte führen eine Liste von Verweisen auf Dozent-Objekte. Letztere hingegen benötigen für die Deputatsverwaltung Verweise auf alle Veranstaltung-Objekte, an denen dann wiederum Deputat-Verweise hängen. So kommt es auch hier zu einer gegenseitigen Benutztbeziehung, die jedoch nicht auflösbar zu sein schien.

Die Umwandlung der „funktionalen Klassen“ in Klassen, die Objekte darstellen, führte – wie bereits erwähnt – zu der Diskussion, ob in dem bisher entwickelten Domänenmodell nicht Teilbereiche zu identifizieren sind, die ihrerseits eine Domäne bilden könnten. Die Idee der Teilbereichsbildung stammt aus der Präsentation der Studienarbeit „Objektorientierter Entwurf und Implementierung eines Systems zur Verzeichnisverwaltung anhand der Kriterien von PETS“[2], die im Rahmen unseres Projektseminars vorgestellt wurde. Dort werden sogenannte *Cluster* als Strukturierungsmittel für die grafische Darstellung des Softwareentwurfs benutzt. Diese Cluster bilden ebenfalls logische Einheiten, die eine sinnfällige Bezeichnung erhalten.[3] So ließen sich viele der von uns modellierten Klassen einem der drei Bereiche „VV-Erstellung“, „Raumbelegung“ und „Deputatsverwaltung“ zuordnen. Das Prüfen auf Überschneidungsfreiheit war als zusätzliche Anforderung in unser Domänenmodell einzubinden. Wir modellierten, diesen Vorgaben entsprechend, die vier (Teil-)Domänen[4] VV-Erstellung, Raumbelegung, De-

[1] Da das Veranstaltung-Objekt an einer von Raum geführten Zeit-Objektliste hängt und nicht etwa direkt an einem Zeit-Objekt, gibt es in Abbildung 5.6-1 keine Verbindung zwischen Zeit und Veranstaltung, wie intuitiv zu vermuten wäre. Diese Beziehung wird erst aus den entsprechenden Klassenkarten ersichtlich.

[2] Stephan Herrmann: Objektorientierter Entwurf und Implementierung eines Systems zur Verzeichnisverwaltung anhand der Kriterien von PETS, Bericht Nr. 1994-11 der Technischen Universität Berlin, 1994

[3] vgl. Herrmann (a.a.O.), Kap. 3.3.1. und 3.3.3.

[4] Im Rückgriff auf die Begriffsbildung aus Kapitel 4.3 können die (Teil-)Domänen auch als Dimensionen bezeichnet werden.

putatsverwaltung und Überschneidungsfreiheit, denen die Klassen – soweit möglich – zugeordnet wurden. In der Domäne Überschneidungsfreiheit wurde eine neue Klasse Priorität geschaffen, die eine Priorisierung der Veranstaltungen bei der Raumbelegung ermöglichen sollte.
Natürlich ließen sich nicht alle Klassen einer der vier (Teil-)Domänen zuordnen, da eine zentrale Klasse wie Veranstaltung von fast allen (Teil-)Domänen benutzt wird. Diese allgemeineren Klassen wurden zwischen den (Teil-)Domänen angesiedelt.
Ergebnis dieses Entwicklungsprozesses war der dritte Entwurf des Domänenmodells, wie er in Abbildung 5.6-2 dargestellt ist. Auffällig ist, daß alle Klassen in diesem Entwurf einen echten Objekt-Charakter haben – es gibt keine „funktionalen Klassen“ mehr. Abbildung 5.6-3 zeigt die Klassenkarten zu diesem Entwurf. Es sind nur diejenigen Attribute und Methoden aufgeführt, die für das fachliche Verständnis des Entwurfs notwendig sind – technische Aspekte wurden völlig außer Acht gelassen.

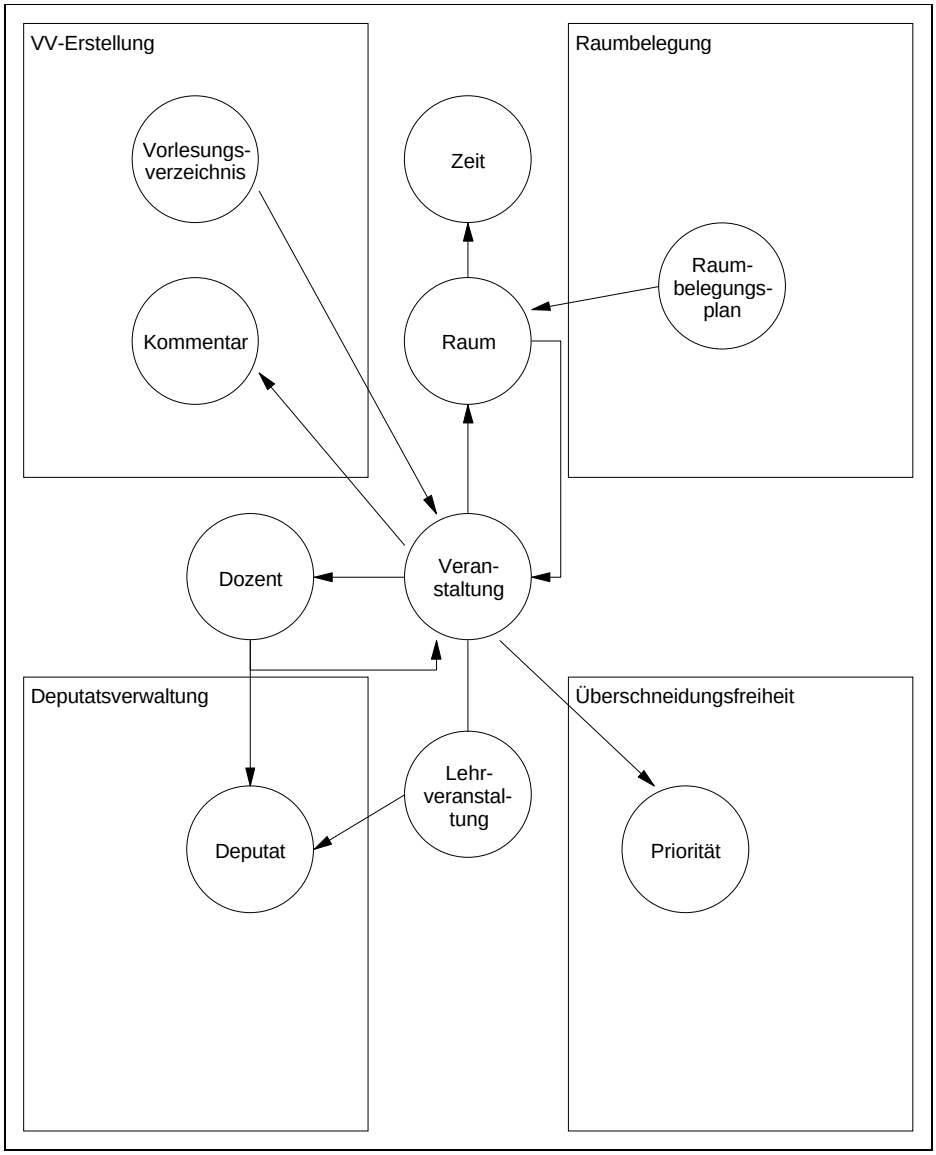

Abbildung 5.6-2: Dritter Entwurf des objektorientierten Domänenmodells, Objektdiagramm

Zeit
• Wochentag • Beginn • Ende

Raum
• Raumnummer • Kapazität $\rightarrow^+$ [Zeit $\rightarrow$ Veranstaltung]
Raumbelegung()

Kommentar
• Kommentartext

Dozent
• Name • Position • Arbeitsbereich $\rightarrow^+$ [Veranstaltung $\rightarrow$ Deputat]

Veranstaltung
• Name $\rightarrow^+$ Dozent $\rightarrow^+$ Raum • Beginn • Periodizität • Hörerzahl $\rightarrow$ Kommentar $\rightarrow$ Priorität

Lehrveranstaltung : Veranstaltung
• Kennummer $\rightarrow^+$ Deputat

Deputat
• Anrechnung • Pflichtstunden
Deputatsverwaltung()

Vorlesungsverzeichnis
$\rightarrow^+$ Veranstaltung

Raumbelegungsplan
$\rightarrow^+$ Raum

Priorität
anwendungsspezifisch implementiert
PrüfeVerträglichkeit(Veranstaltung)

Abbildung 5.6-3: Dritter Entwurf des objektorientierten Domänenmodells, Klassenkarten

Diesen Entwurf diskutierten wir wiederum mit allen Seminarteilnehmern.

5.7 Vierter Entwurf: Autor-Kritiker-Zyklus II / Überarbeitung der Klassen

Bemängelt wurde die starre Beschreibung des zeitlichen Verhaltens von Veranstaltungen. Bisher wurde davon ausgegangen, daß Veranstaltungen wöchentlich stattfinden, so daß für jeden Wochentag, an dem die Veranstaltung angesetzt ist, ein entsprechendes Raum-Objekt mit anhängendem Zeit-Objekt von der Veranstaltung geführt wird. Einmalige Veranstaltungen, wie z.B. Gastvorträge, oder Blockveranstaltungen können in diesem Modell nur sehr aufwendig beschrieben werden. Das Zeit-Objekt mußte also entsprechend flexibler gestaltet werden.
Eine grundsätzliche Überarbeitung sollte auch die Klasse Veranstaltung erfahren. So hatte

bisher jede Veranstaltung einen Dozenten, obwohl durchaus auch Nicht-Dozenten als Veranstalter denkbar sind. Wir führten als Antwort darauf eine neue Klasse Veranstalter ein. Ein Dozent kann als Spezialisierung eines Veranstalters aufgefaßt werden. Folglich ist Dozent eine von Veranstalter abgeleitete Klasse, wie Abbildung 5.7-1 zeigt:

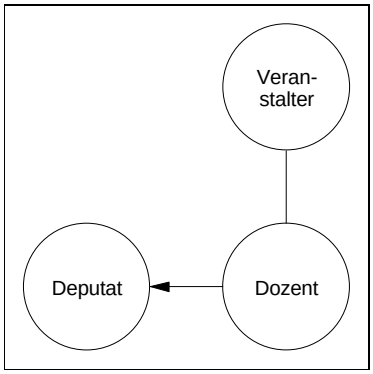

Abbildung 5.7-1: Einführung der Klasse Veranstalter

In unserem Domänenmodell wurden alle Veranstaltungen, die keine Lehrveranstaltung waren, als Objekte der allgemeinen Klasse Veranstaltung betrachtet. Somit stehen Veranstaltung- und Lehrveranstaltung-Objekte aus Sicht der Klassenhierarchie auf unterschiedlichen Abstraktionsebenen, obwohl beide konkrete Ausprägungen einer Veranstaltung sind. Als konzeptionell saubere Lösung bot sich die Abstraktion der Klasse Veranstaltung an: Von Veranstaltung selbst als abstrakte Klasse kann kein Objekt direkt erzeugt werden, sondern nur von Spezialisierungen dieser Klasse. Als Spezialisierungen wurden die Klassen Allgemeine Veranstaltung und Lehrveranstaltung modelliert.
Der Veranstaltung-Klassenbaum beinhaltet Klassen für Veranstaltungen, die in unterschiedlichen Kontexten eingesetzt werden können. Aus diesem Grund stehen sie auch zwischen den vier (Teil-)Domänen. Im Rahmen der VV-Erstellung reichen die Attribute solcher Veranstaltungen jedoch nicht unbedingt aus, um jede Veranstaltung eindeutig zu identifizieren – eine unbedingte Voraussetzung für diesen Aufgabenbereich. Charakteristisch für die Veranstaltungen in Vorlesungsverzeichnissen ist eine Klassifikation. In Hamburg z.B. werden Veranstaltungen durch die LV-Nummer klassifiziert, in Berlin durch die Lehrveranstaltungskennziffer. Für die Domäne VV-Erstellung gibt es also spezielle Ausprägungen der Klassen Allgemeine Veranstaltung und Lehrveranstaltung: Klassifizierte Allgemeine Veranstaltung und Klassifizierte Lehrveranstaltung erben zusätzlich von der Klasse Klassifikation, in der die konkrete Klassifizierung für die jeweilige Anwendung spezifisch implementiert ist. Somit konnten Spezialisierungen der Klasse Veranstaltung direkt der (Teil-)Domäne VV-Erstellung zugeordnet werden. Abbildung 5.7-2 zeigt die neue Modellierung der Klasse Veranstaltung.

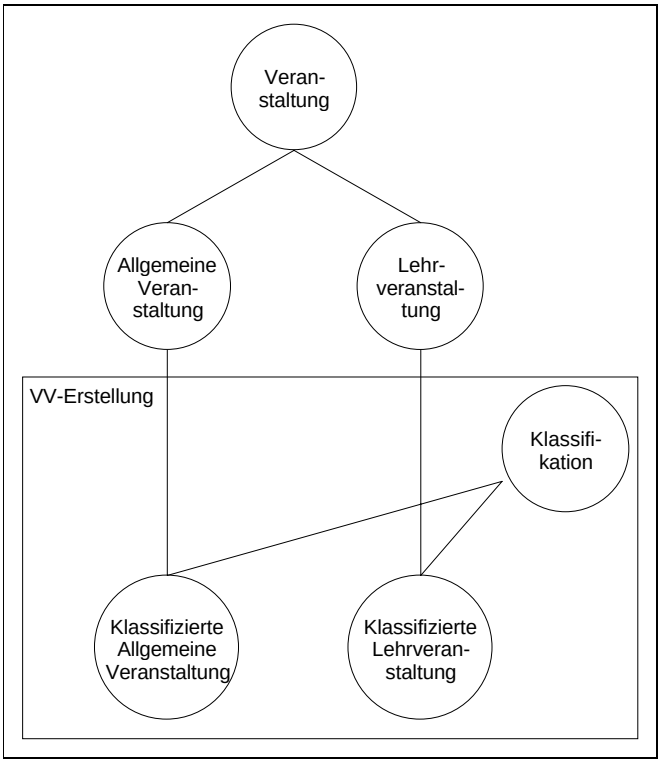

Abbildung 5.7-2: Neue Modellierung der Klasse Veranstaltung

Der (Teil-)Domänenname Überschneidungsfreiheit mißfiel den Seminarteilnehmern, da er nicht wie die übrigen drei Domänennamen eine Funktion beschrieb. Aus diesem Grund wurde er auf Belegungskonfliktlösung geändert.
Außerdem fiel uns auf, daß es in unserem Domänenmodell nicht möglich war, eine neue Lehrveranstaltung anzumelden. Dieser Mangel konnte durch Schaffung der neuen (Teil-)Domäne LV-Anmeldung behoben werden.
Diese Diskussion und die anschließend notwendig gewordenen Änderungen an einzelnen Klassen führten zum vierten Entwurf des objektorientierten Domänenmodells, wie in Abbildung 5.7-3 und Abbildung 5.7-4 als Objektdiagramm bzw. Klassenkarten dargestellt. Dieser vierte Entwurf enstand am Ende des Semesters und konnte den Seminarteilnehmern deshalb nicht mehr vorgestellt werden.
Aus Gründen der Übersichtlichkeit mußten im Objektdiagramm die einzelnen (Teil-)Domänen umplaziert werden. Die strichlierte gerichtete Kante von der (Teil-)Domäne LV-Anmeldung zur Klasse Veranstaltung soll andeuten, daß in dieser (Teil-)Domäne neue Objekte dieser Klasse (bzw. einer abgeleiteten Klasse) erzeugt werden.
Weiterhin überarbeitet wurde der Raumbelegungsplan. Dieser führt nun eine Liste von Raum-/Zeit-Paaren. Die Raum-Klasse wurde von der Veranstaltung-Liste befreit, um ihrem eigentlichen Charakter – der Repräsentation eines Raumes – besser gerecht zu werden. Von der Klasse Zeit gibt es die Spezialisierungen Zeitpunkt, die ein Datum mit einschließt, und Periode mit einer Periodizität.
Die Deputatsverwaltung wurde vom Rest des Modells entkoppelt, da sie z.B. in Berlin nicht benötigt wird. Jedes Deputat-Objekt enthält einen Verweis auf ein Lehrveranstaltung- und ein Dozent-Objekt.

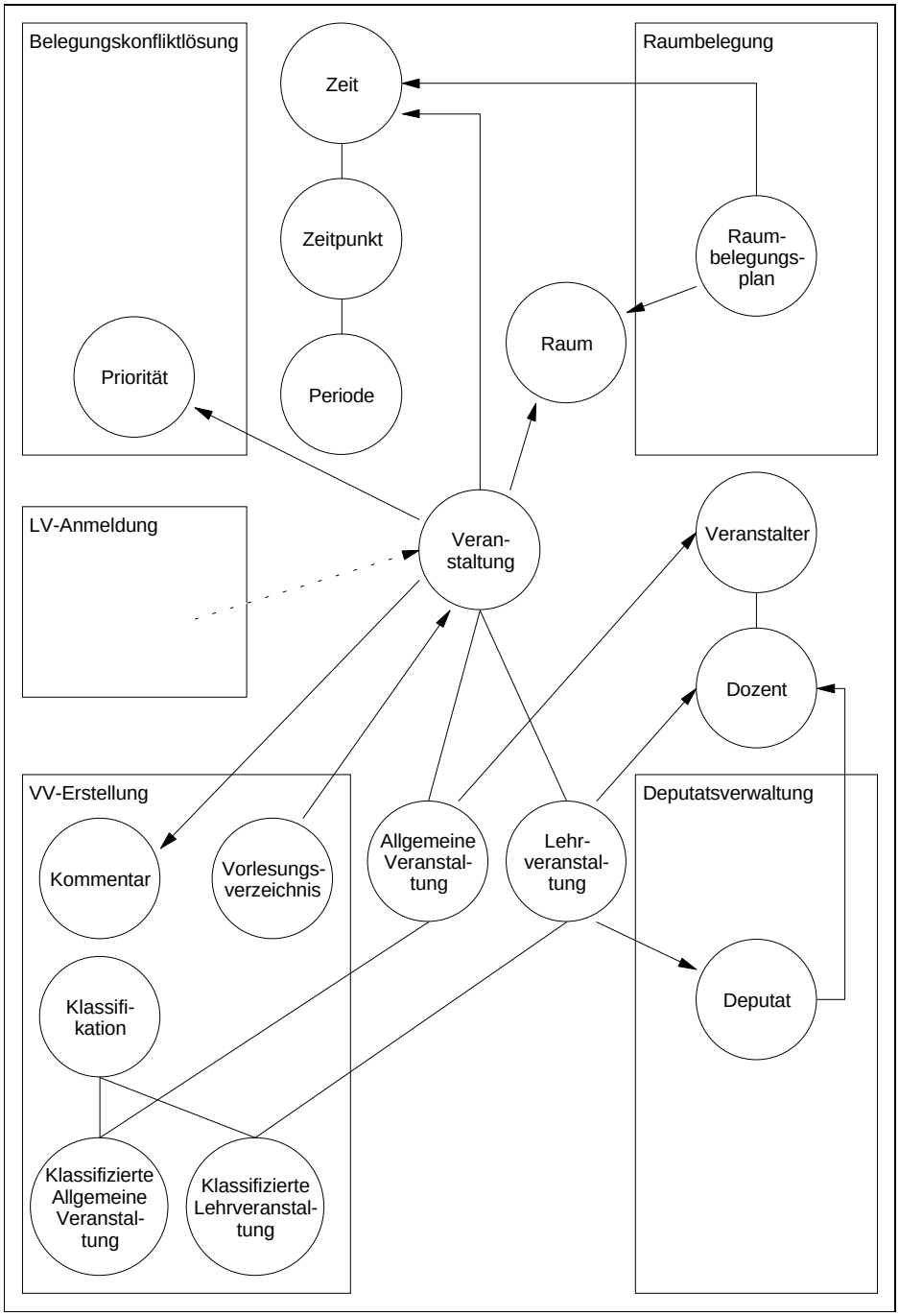

Abbildung 5.7-3: Vierter Entwurf des objektorientierten Domänenmodells, Objektdiagramm

Zeit
• Beginn • Ende

Zeitpunkt : Zeit
• Datum

Periode : Zeit
• Periodizität

Raum
• Raumnummer • Kapazität

Kommentar
• Kommentartext

Veranstalter
• Name

Dozent : Veranstalter
• Position • Arbeitsbereich

Veranstaltung (abstrakte Klasse)
• Titel $\rightarrow^+$ [Raum $\rightarrow$ Zeit] • Hörerzahl $\rightarrow$ Kommentar $\rightarrow$ Priorität

Allgemeine Veranstaltung : Veranstaltung
$\rightarrow^+$ Veranstalter

Lehrveranstaltung : Veranstaltung
• Kennummer

Klassifikation
• Kennummer

Klassifizierte Allgemeine Veranstaltung : Allgemeine Veranstaltung, Klassifikation

Klassifizierte Lehrveranstaltung : Lehrveranstaltung, Klassifikation

Vorlesungsverzeichnis
$\rightarrow^+$ Veranstaltung
FügeVeranstaltungEin() EntferneVeranstaltung() ErstelleVorlesungsverzeichnis ()

Raumbelegungsplan
$\rightarrow^+$ [Raum $\rightarrow$ Zeit]
ErstelleRaumbelegungsplan()

Deputat
$\rightarrow$ Dozent $\rightarrow$ Lehrveranstaltung • Anrechnung • Pflichtstunden
Deputatsverwaltung()

Priorität
kundenspezifisch implementiert
PrüfeVerträglichkeit(Veranstaltung)

Abbildung 5.7-4: Vierter Entwurf des objektorientierten Domänenmodells, Klassenkarten

5.8 Ergebnisse der Domänenmodellierung

Wir haben versucht, für den Anwendungsbereich der Erstellung eines Vorlesungsverzeichnisses ein objektorientiertes Domänenmodell zu entwickeln. Nach Einarbeitung in die Konzepte der Objektorientierung und Auswahl geeigneter Darstellungsmittel (Objektdiagramme, Klassenkarten) begannen wir, vornehmlich in den Aufgabennetzen Objekte des Anwendungsbereiches zu identifizieren. Diese Objekte bildeten die Grundlage für den ersten Modellentwurf, der in mehreren Autor-Kritiker-Zyklen diskutiert und jeweils anschließend überarbeitet wurde. Die evolutionäre Entwicklung des Modells, wie sie auch von STEPS[1] propagiert wird, hat sich als effektive und erfolgreiche Vorgehensweise erwiesen.
Als problematisch erwies sich die Entwicklung des objektorientierten Domänenmodells aus den Aufgabennetzen heraus. Die Aufgabennetze sind zwar objektbasiert, doch die Objekte (in STEPS „Informationen“ oder „Objekte“ bzw. „Informationsstrukturen“ genannt[2]) dienen ausschließlich als Medium für die Kommunikation und Interaktion zwischen funktionellen Rollen. Strukturen, die für den objektorientierten Entwurf wichtig sind, wie Beziehungen der Objekte untereinander oder Objekthierarchien bzw. Vererbungsbeziehungen, sind den Aufgabennetzen nicht zu entnehmen.
Die wohl wichtigste Entdeckung war, daß die von uns betrachtete Domäne „Erstellung eines Vorlesungsverzeichnisses“ offenbar weitere (Teil-)Domänen enthielt. Bei der Diskussion der Klassenkarten wurde von einigen Seminarteilnehmern der Vorschlag gemacht, aus dem Domänenmodell Werkzeuge und Materialien nach der *Werkzeug-Material-Metapher*[3] (im arbeitsbereichsinternen Sprachgebrauch auch „*WAM-Methode*“[4] genannt) zu entwickeln. Dieses Leitbild war uns zum damaligen Zeitpunkt noch unbekannt, so daß wir auf eine entsprechende Umsetzung dieses Vorschlags verzichten mußten. Wir erkannten jedoch, daß die Aufgabengebiete der von uns identifizierten (Teil-)Domänen durch jeweils eigene Werkzeuge bearbeitet werden könnten. In dem nun folgenden Kapitel sollen Ideen für eine Umsetzung des Domänenmodells nach der WAM-Methode aufgezeigt werden.

5.9 Ideen für eine Umsetzung des Modells nach der Werkzeug-Material-Metapher

Im folgenden soll in Ansätzen versucht werden, ausgehend vom vierten Entwurf des objektorientierten Domänenmodells ein objektorientiertes Klassenmodell zu entwickeln, das der Werkzeug-Material-Metapher genügt. Dabei sollen zum einen die Vorgehensweise beim objektorientierten Entwurf und zum anderen diese Metapher in Anwendung auf unser Fallbeispiel erklärt werden.
Objektorientierung, so wie wir sie verstehen und angewendet haben, bedeutet allgemein, die „relevanten Gegenstände und Begriffe eines Anwendungsbereichs fachlich zu beschreiben [...] [und] [...] diesen fachlichen Entwurf ohne Modellbruch in einen technischen Entwurf zu überführen“.[5] Was die Objektorientierung jedoch nicht löst ist die Frage, „wie Gegenstände bei der Analyse eines Anwendungsbereichs als relevant erkannt werden können und vor allem, wie das angestrebten System und die zukünftigen Arbeitsformen aussehen können“.[6] Hier kann eine Entwurfsmetapher Abhilfe schaffen, die dem Entwickler bei der Entwicklung des Entwurfs der

[1] vgl. Floyd (a.a.O.)
[2] vgl. Floyd (a.a.O.), Kap. 3.3.2
[3] vgl. Kilberth et al. (a.a.O.), Kap. 2.2
[4] Das Akronym WAM steht unseres Wissens nach für „Werkzeug-Aspekt/Automat-Material“ und soll im folgenden abkürzend für diese Methode bzw. die Werkzeug-Material-Metapher stehen.
[5] vgl. Kilberth et al. (a.a.O.), S. 23
[6] vgl. Kilberth et al. (a.a.O.), S. 23

(Software-)Anwendung aus den bei der Analyse gewonnenen Informationen heraus eine Hilfestellung bietet.
Als Leitbild verwendet die WAM-Methode den „Arbeitsplatz für qualifizierte menschliche Tätigkeiten“[1], in dessen Rahmen als Entwurfsmetapher der „Umgang mit Werkzeug und Material“[2] zum Einsatz kommt. Die Ausprägungen dieser Metapher sind Werkzeug- und Materialklassen, die sich aus dem Anwendungsbereich heraus fachlich motivieren lassen. Die Tatsache, daß ein Werkzeug und ein Material „zueinander passen“, wird durch eine Aspektklasse ausgedrückt, die als Schnittstelle dem Werkzeug diejenigen Eigenschaften des Materials beschreibt, die für das Werkzeug in diesem Kontext relevant sind.[3]
Den Schwerpunkt der Klassenentwürfe bilden die Materialien des Anwendungsbereichs, deshalb soll auch mit der Zuordnung zu Materialien begonnen werden.

5.9.1 Materialien

Materialien, die im Zusammenhang mit der Tätigkeit[4] „Erstellung eines Vorlesungsverzeichnisses“ stehen, sind im wesentlichen bereits bei der Domänenanalyse identifiziert worden. Allerdings haben die dort entwickelten Klassen keinen eigentlichen Materialcharakter im Sinne von Arbeitsgegenständen. Man denke z.B. an die Klasse Lehrveranstaltung, die zwar ein Objekt beschreibt, daß im Kontext der Vorlesungsverzeichniserstellung verwendet wird, als konkretes Material jedoch nur in Form einer Lehrveranstaltungsbeschreibung vorkommt. Die WAM-Methode versucht zunächst, die Materialien des Anwendungsbereiches zu identifizieren und in Klassen zu modellieren. Dieser Tatsache ist im folgenden Rechnung zu tragen. So müssen alle Klassen, die Objekte beschreiben, denen im Anwendungsbereich konkrete Materialien entsprechen, durch die entsprechenden Materialklassen ersetzt werden.
Vergleicht man nun das Hamburger Modell mit dem Berliner Modell, so stellt man sehr schnell fest, daß in beiden Modellen für gleiche Tätigkeiten unterschiedliche Materialien oder aber gleiche Materialien mit unterschiedlichen Bezeichnungen verwendet werden. In Hamburg z.B. werden die Lehrveranstaltungen in Lehrtableaus erfaßt, während in Berlin neue Lehrveranstaltungen über Erfassungsbelege angemeldet werden und für eine bereits angemeldete Lehrveranstaltung ein Lektionsbogen existiert. Will man also ein System nach der WAM-Methode entwickeln, so müßte man sich auf einen speziellen Anwendungsfall festlegen, um die dort verwendete Fachsprache der Anwender in den Entwurf übernehmen zu können. Diese Adaption der Fachsprache der Anwender ist eines der wichtigsten Konzepte der Anwenderpartizipation, wie sie auch in STEPS propagiert wird.[5] Allgemeine Lösungen für mehrere Anwendungsfälle (eines Anwendungsgebietes), wie sie die Domänenanalyse zu entwerfen erlaubt, scheinen demnach mit der WAM-Methode nicht erstellt werden zu können.
Erfahrungen aus konkreten Projekten, in denen objektorientierte Software nach der WAM-Methode entwickelt wurde, zeigen jedoch, daß die Methode durchaus für fallübergreifende Softwarelösungen geeignet ist. So wird beispielsweise im RWG-Bankenprojekt[6] ein integriertes Sachbearbeitungssystem für die württembergischen Volks- und Raiffeisenbanken entwikkelt. Das System bietet eine einheitliche Oberfläche, berücksichtigt aber trotzdem die unterschiedlichen Organisationsstrukturen der einzelnen Banken. Ein solches Projekt kann nur erfolgreich sein, wenn Fragen der Begriffsbildung unter expliziter Einbeziehung der Benutzer geklärt werden und gemeinsam Abstraktionen geschaffen werden, die eine partielle Vereinheit-

[1] vgl. Kilberth et al. (a.a.O.), S. 24
[2] vgl. Kilberth et al. (a.a.O.), S. 24
[3] Genauere Ausführungen und Beispiele zu diesem Thema sind bei Kilberth et al. zu finden.
[4] Statt „Domäne“ wird hier der Begriff „Tätigkeit“ benutzt, um die Verwendung der WAM-Terminologie konsistent zu halten.
[5] vgl. Floyd (a.a.O.), Kap. 1.6
[6] vgl. Kilberth et al. (a.a.O.), Kap. 8.5.2

lichung erlauben. Einen ähnlichen Weg geht die Domänenanalyse, so daß ein Domänenmodell letztendlich doch unter Anwendung der WAM-Methode in ein Software-System umgesetzt werden kann. Dazu bedarf es jedoch der Mitwirkung von Benutzern aus den einzelnen konkreten Anwendungsfällen, um Abstraktionen zu finden, die für möglichst alle Beteiligten vertretbar sind bzw. logisch erscheinen. Da es uns an solchen Benutzern mangelt und ausführliche Benutzer-Interviews zudem den Rahmen dieser Studienarbeit sprengen würden, beschränken wir uns bei der weiteren Modellierung auf das Hamburger Modell, da für dieses mehrere Materialien in natura vorliegen[1]. Aus diesen und aus den bei der Domänenmodellierung gefundenen Objekten lassen sich folgende Materialien identifizieren:

- Lehrveranstaltungsbeschreibung
  Beschreibt eine Lehrveranstaltung am Fachbereich Informatik der Universität Hamburg für ein Semester durch Angabe von Veranstaltungsart, Titel, Dozenten, Semesterwochenstunden und Teilnehmerzahl.
- Deputatsformular ProfessorInnen
  Beschreibt für alle Professoren eines Arbeitsbereiches am Fachbereich Informatik der Universität Hamburg die Pflichtveranstaltungen in Grund- und Hauptstudium mit der Anzahl anzurechnender Semesterwochenstunden.
- Deputatsformular Wimis
  Beschreibt für alle wissenschaftlichen Mitarbeiter eines Arbeitsbereiches am Fachbereich Informatik der Universität Hamburg die Pflichtveranstaltungen mit der Anzahl anzurechnender Semesterwochenstunden.
- Lehrtableau
  Beschreibt alle Lehrveranstaltungen eines Arbeitsbereiches am Fachbereich Informatik der Universität Hamburg für ein Semester durch Angabe von Veranstaltungsart, Titel, Dozenten, Semesterwochenstunden und Teilnehmerzahl.
- Stupla-Lehrveranstaltungsbeschreibung
  Beschreibt eine Lehrveranstaltung am Fachbereich Informatik der Universität Hamburg für ein Semester innerhalb des Stundenplanprogramms Stupla. Die Beschreibung enthält Lehrveranstaltungsnummer, Veranstaltungsart, Titel, Dozenten und Semesterwochenstunden und geht in das Vorlesungsverzeichnis ein.
- Vorlesungsverzeichnis
  Beschreibt alle Lehrveranstaltungen aller Fachbereiche der Universität Hamburg für ein Semester durch Angabe von Veranstaltungsart, Titel, Dozenten, Semesterwochenstunden und Teilnehmerzahl.
- Lehrveranstaltungs-Ankündigung
  Eine kommentierte Beschreibung einer Lehrveranstaltung am Fachbereich Informatik der Universität Hamburg für ein Semester, die in das kommentierte Vorlesungsverzeichnis eingeht.
- Kommentiertes Vorlesungsverzeichnis
  Beschreibt und kommentiert alle Lehrveranstaltungen am Fachbereich Informatik der Universität Hamburg für ein Semester auf Grundlage der Lehrveranstaltungs-Ankündigungen.
- Raumbelegungsplan
  Führt für jeden für Lehrveranstaltungen genutzten Raum am Fachbereich Informatik der Universität Hamburg für das laufende Semester die Belegung durch Lehrveranstaltungen unter Angabe von Lehrveranstaltungsnummer, Titel und Dozenten auf.

[1] siehe Anhang F

5.9.2 Werkzeuge

Die Materialien sollen im Rahmen von WAM von Software-Werkzeugen bearbeitet werden, in Anlehnung an die natürliche Bearbeitung von Arbeitsgegenständen mit geeigneten Werkzeugen. Als Vorbild für die zu entwerfenden Software-Werkzeuge dienen jedoch nicht unmittelbar diese im Anwendungsbereich vorhandenen Werkzeuge, sondern vielmehr „bestimmte charakteristische Arbeitstätigkeiten, wie etwa das Sortieren von Karteikarten oder das Durchsuchen von Dokumentationen. Bei der Entwicklung eines Anwendungsmodells kommen daher neben einigen bereits vorhandenen Werkzeugen weitere hinzu.“[1]
Beim Entwurf der Werkzeuge kann man sich grob an den (Teil-)Domänen des Domänenmodells orientieren, da diese einzelne Arbeitskontexte voneinander trennen. Allerdings steht nicht unbedingt eine Domäne für genau ein Werkzeug, wie am Beispiel der (Teil-)Domäne VV-Erstellung leicht deutlich gemacht werden kann: Man benötigt einen Editor für Veranstaltungen, die in das Vorlesungsverzeichnis eingehen sollen, aber mindestens auch einen Browser, um eine Vorlesung im elektronischen Vorlesungsverzeichnis wiederzufinden.
Die Werkzeuge sollten im Umgang mit den zu bearbeitenden Materialien möglichst viel vom natürlichen Umgang im Anwendungsbereich übernehmen. So sollten für alle Formulare, wie z.B. das Deputatsformular, elektronische Pendants geschaffen werden (siehe Abschnitt „Materialien“), die mit einem Deputatsformular-Editor und einen Deputatsformular-Browser bearbeitet werden können.
Die Raumbelegung hat als Ergebnis einen Raumbelegungsplan, auf den der Ersteller im Idealfall möglichst wenig Einfluß haben muß, da die Informationen aus der Lehrveranstaltungsbeschreibung und die Angabe einer Priorität ausreichen sollten, um die Räume automatisch zu belegen. Natürlich muß die Möglichkeit einer manuellen Umbelegung gegeben sein, doch die grundsätzliche Funktionalität erfordert weniger ein Werkzeug denn einen Automaten, der – evtl. nach Einstellen einiger Parameter – automatisch im Hintergrund seine Arbeit verrichtet.

5.9.3 Automaten

Für genau die oben genannten Zwecke sieht auch die WAM-Methode einen Automaten vor. Das Hauptmerkmal eines Automaten gegenüber einem Werkzeug ist die Aktivität im Hintergrund, d.h. „er führt über einen längeren Zeitraum Aktionen auf der Basis eines vorgegebenen Algorithmus ‘automatisch’ durch“.[2] Ergänzend zu einem Raumbelegungsautomaten wäre auch ein Raumbelegungs-Editor sinnvoll, der ersteren anstößt und nach Beendigung des Automaten dem Anwender die Ergebnisse der automatischen Raumbelegung präsentiert, so daß dieser noch Änderungen vornehmen kann.

5.9.4 Aspekte

Werkzeuge können durchaus mehrere Materialien bearbeiten. Ebenso ist es denkbar, daß ein Material von verschiedenen Werkzeugen bearbeitet werden kann. Dieser Zusammenhang ist jedoch nicht beliebig. Was man braucht, ist ein Beschreibungsmittel für das „Zueinanderpassen“ von Werkzeugen und Materialien. Ein solches Beschreibungsmittel sind Aspektklassen, die als Schnittstellen dem Werkzeug alle in Zusammenhang mit dem zu bearbeitenden Material relevanten Eigenschaften bekannt machen. So könnte ein Werkzeug Drukker den Aspekt bzw. die Aspektklasse Druckbar benutzen, um Materialien wie Raumbelegungsplan oder Kommentiertes Vorlesungsverzeichnis zu drucken, die (i.d.R. über Mehrfachvererbung) von Druckbar erben. Neben solch allgemeinen Aspektklassen, die man in nahezu jedem Software-System identifizieren könnte, gibt es auch anwendungsbereichsspezifische

[1] vgl. Kilberth et al. (a.a.O.), S. 29
[2] vgl. Kilberth et al. (a.a.O.), S. 77

Aspektklassen. So wäre ein Aspekt Priorisierbar für Lehrveranstaltungsbeschreibungen denkbar. Der Raumbelegungsautomat könnte dann Priorisierbar nutzen, um bei der Raumbelegung auftretende Belegungskonflikte zu lösen. Genauere Ausführungen zum Thema Aspektklassen finden sich bei Kilberth et. al.[1]

Zusammenfassend kann man sagen, daß objektorientierte Domänenmodelle durchaus als Ausgangspunkt für die Anwendungsmodellierung nach der Werkzeug-Material-Metapher geeignet sind. Grundsätzlich hat die WAM-Modelle jedoch keine so allgemeine Ausrichtung wie Domänenmodelle, da sie auf die Modellierung eines konkreten Anwendungsfalles zugeschnitten ist. Das wird vor allem daran deutlich, daß die Sprache der Anwender im konkreten Anwendungsfall in den Entwurf übernommen wird. Domänenmodelle müssen von solchen Details abstrahieren, da sie die Essenz aus der Betrachtung mehrerer verschiedener Beispiele und somit verschiedener Anwendungsfälle sind. Erfahrungen aus der Praxis (z.B. das RWG-Bankenprojekt) zeigen jedoch, daß durch Diskussion mit den Benutzern und gemeinsame Begriffsabstraktion Anwendungsmodelle nach der Werkzeug-Material-Metapher entwickelt werden können, die sich in mehr als einem Anwendungsfall (eines bestimmten Anwendungsbereiches) einsetzen lassen.

[1] vgl. Kilberth et al. (a.a.O.), Kap. 2.2.4

6 Kritische Schlußbemerkungen

In diesem Kapitel sollen die wichtigsten Arbeitsgrundlagen, die Erkenntnisse der Studienarbeit und das Arbeitsumfeld kritisch besprochen werden.

Durch den großen Zeitraum von einem Jahr zwischen dem Projektseminar und dem Beginn der Studienarbeit waren uns einige Fakten und Zusammenhänge nicht mehr präsent. Gute Mitschriften sind zwar hilfreich, können jedoch nie alle benötigten Informationen enthalten. So mußten wir uns diese fehlenden Informationen noch einmal erschließen, was ein zeitaufwendiger Prozeß war.

Sehr unterschiedliche Definitionen selbst grundlegender Begriffe der DA erschwerten die Einarbeitung anhand der Literatur. An dieser Stelle sei nochmals darauf hingewiesen, daß selbst die Begriffe Domäne und Problemdomäne – so wie wir sie verwenden – nicht allgemein anerkannt sind.

Die in Kapitel 3 verwendeten Aufgabennetze sind schwer zu erstellen und zu überblicken, wenn die Beziehungen zwischen den einzelnen funktionellen Rollen komplizierter werden. Wir haben viel Zeit darauf verwendet, die einzelnen graphischen Elemente richtig anzuordnen, da ein Aufgabennetz auch noch auf eine DIN A4-Seite passen sollte. Ab einer gewissen Informationsmenge wird deshalb häufig über mehrere Stufen verfeinert. Damit wird die Abstimmung der einzelnen Verfeinerungsstufen ein großes Problem. Fehler in den Aufgabennetzen sind jetzt schwer zu entdecken.

Wir haben nur zwei Beispiele innerhalb der Domäne VV–Erstellung untersucht. Lediglich für Hamburg hatten wir auch genügend Zugriff auf Dokumente und Domänenexperten. Für die Berliner VV–Erstellung standen uns nur wenige Dokumente und Domänenexperten zur Verfügung. Diese beiden Beispiele wurden im Projektseminar als Arbeitsgrundlage eingeführt und in unserer Studienarbeit benutzt. Jedes weitere Beispiel hätte den Rahmen der Studienarbeit gesprengt. Solche Einschränkungen sind innerhalb einer Studienarbeit kaum zu umgehen, werfen allerdings die Frage nach der Validität des Domänenmodells auf. Zu unserer eigenen Überraschung ist unserer erstes komplett selbst erstelltes Domänenmodell trotz dieser Einschränkungen sehr zusammenhängend und deckt sowohl Berlin als auch Hamburg ab.

Eine der interessantesten Beobachtungen haben wir durch den Vergleich unserer Einschätzung über die Machbarkeit der DA am Anfang und am Ende unserer Arbeit in der Projektgruppe gemacht. Während wir am Anfang der DA die Machbarkeit als sehr gering einschätzten, waren wir am Ende überrascht, ein so allgemeines und trotzdem beide Beispiele so gut abdeckendes Kernmodell gefunden zu haben. Unsere anfängliche Skepsis ist dadurch zu begründen, daß wir uns gerade mit den Informationen über Berlin und Hamburg auseinandergesetzt hatten und wir noch keine Erfahrung mit der Erstellung eines Domänenmodells sammeln konnten. So sahen wir viele Unterschiede in Berlin und Hamburg (Begrifflichkeiten, Abläufe usw.) und hatten keine Informationen über ein Verfahren, wie diese Unterschiede in einem Kernmodell zu vereinen wären. Zudem wurden die Berliner Aufgabennetze von einer anderen Teilgruppe unserer Arbeitsgruppe erstellt und unterschieden sich so – neben den fachlich motivierten Unterschieden – graphisch und im Aufbau stark von unseren Hamburger Aufgabennetzen. Während der nun folgenden Domänenbildung (siehe Kapitel 5) waren vor allem folgende Erkenntnisse für unsere letztendlich positive Beurteilung der Möglichkeiten der DA wichtig:

Die „funktionalen Klassen“ (siehe S. 47) wurden als nicht als Objekte darstellbar erkannt. Aus diesen „funktionalen Klassen“ entwickelten sich die Teildomänen, was eine bessere Gliederung der Domäne „VV-Erstellung“ ermöglicht und auch die Möglichkeit eröffnet, einzelne Teildomänen wegzulassen, falls diese nicht benötigt werden. So lassen sich starke Unterschiede in den analysierten Beispielen in einem Kernmodell vereinen (hier: Deputatsverwaltung, siehe Abbildung 5.6-2). Durch die Unterteilung der Domänen in Teildomänen lassen sich Klassen identifizieren, die so allgemein sind, daß sie zwischen den Teildomänen angesiedelt werden, da sie von mehreren Teildomänen benötigt werden. So erkennt man unterschiedlich starke Abhängigkeiten zwischen den einzelnen Teildomänen (siehe Abbildung 5.7-3), da die Teildomänen um so abhängiger voneinander sind, je mehr allgemeine Klassen sie gemeinsam benutzen.

Die vier Entwürfe unseres Kernmodells wurden innerhalb von zwei Monaten mit der STEPS-Methodik entwickelt. Dabei hat sich STEPS gut bewährt. Wir haben die Kernmodelle in unserer Teilgruppe, in der Arbeitsgruppe und im Plenum des Projektseminares mehreren Autor-Kritiker-Zyklen unterworfen. (Die Autor-Kritiker-Zyklen waren in unserer Teilgruppe natürlich wesentlich häufiger als im Plenum.) Diese Vorgehensweise hat sich sehr bewährt, da so unterschiedlichste Ideen in unsere Entwürfe einfließen konnten; Fehler oder Dissonanzen im Modell oder im zugrundeliegenden Konzept wurden schneller erkannt.

Anhang A Glossar Hamburg

AB-Lehrplanbeauftragter

Der AB-Lehrplanbeauftragte ist als Verwaltungsinstanz zwischen den Lehrveranstaltern seines Arbeitsbereiches und den für die Planung des Fachbereichs zuständigen Personen (Raumplaner, KVV-Planer und Lehrplanbeauftragter) bestimmt. Er übernimmt Weitergabe- und Kontrollfunktionen aller ihm zukommenden Dokumente und koordiniert die seinem Arbeitsbereich zugehörigen Räumlichkeiten.

AB-Lehrtableau

Inhalt:
Veranstaltungsart, Titel, Semesterstundenzahl, Kern-/Vertiefungsgebiet, Teilnehmerzahl (TN), Veranstalter
Umgang:
↳AB-Lehrplanbeauftragter erhält dieses Formular von dem ↳Lehrplanbeauftragten, läßt von den ↳Lehrveranstaltern die entsprechenden Daten eintragen und sendet dieses Formular zurück.

Absprache

Telefon- und Schriftverkehr zur Absprache von Raum- und Zeitplanung zwischen ↳Raumplaner und ↳AB-Lehrplanbeauftragtem.

Anforderungsschreiben

Inhalt:
siehe Anhang E
Umgang:
Begleitschreiben und Aufforderung zum Kommentieren ihrer Lehrveranstaltungen an die entsprechenden Veranstalter.

Deputatarchiv

Ordner aller ↳Deputatsformulare, verwaltet durch den ↳Lehrplanbeauftragten.

Deputatsformular

Inhalt:
Name, Veranstaltungsnummern, Anrechnung für Deputat (siehe Anhang D)
Umgang:
↳AB-Lehrplanbeauftragter erhält dieses Formular von dem ↳Lehrplanbeauftragten, läßt von den ↳Lehrveranstaltern die entsprechenden Daten eintragen und sendet dieses Formular zurück. Es dient zur Kontrolle der geforderten Deputatsrichtlinien.

Deputatsverwaltung

Der ↳Lehrplanbeauftragte übernimmt die Deputatsverwaltung. Er überprüft, ob jeder Veranstalter eine bestimmte Mindeststundenzahl an Veranstaltungen im Semester abhält.

Druckformatvorlagen

Eine Word-Druckformatvorlage und eine Beschreibung, wie diese zu benutzen ist, wird vom ↳KVV-Planer versendet, um einheitlich kommentierte Lehrveranstaltungsbeschreibungen zu erhalten. Zum genauen Aufbau siehe Anhang E.

Fachbereichsrat

Der Fachbereichsrat hat im Zusammenhang mit der Erstellung des Vorlesungsverzeichnisses die Aufgabe, das ↳Lehrtableau zu verabschieden. Die ist Grundvoraussetzung für die Erstellung des ↳Vorlesungsverzeichnisses.

FBR
↳Fachbereichsrat

Kommentierte LV-Beschreibung
Ergebnis der Aufforderung zum Kommentieren der ↳Lehrveranstaltungen durch den ↳KVV-Planer. Es handelt sich entweder um eine Datei, die mit Hilfe der ↳Druckformatvorlage erstellt wurde, oder ein inhaltlich gleichwertiges Dokument.

Kommentiertes Vorlesungsverzeichniss
Endergebnis aller Arbeitsschritte. Käuflich zu erwerben. Zum genauen Aufbau siehe Anhang E

KVV-Planer
Der KVV-Planer hat die Aufgabe, das kommentierte Vorlesungsverzeichnis zu erstellen. Er beginnt seine Tätigkeit ca. 4 Wochen vor Drucklegung. Zu diesem Zeitpunkt sind alle planerischen Tätigkeiten des ↳Lehrplanbeauftragten und des ↳Raumplaners abgeschlossen. Er fordert die Veranstalter zum Kommentieren auf (↳Anforderungsschreiben) und liefert eine vorgefertigte ↳Druckformatvorlage. Die ↳kommentierten LV-Beschreibungen werden über einen PC mit dem Textverarbeitungsprogramm Word überarbeitet (↳Layoutabgleich) und zu einer ↳KVV-Text(datei) zusammengeführt. Diese wird einem Verlag zum Druck und Vertrieb weitergeleitet.

KVV-Text(datei)
MS-Word-Datei, die als Grundlage für das durch den Verlag gedruckte ↳Kommentierte Vorlesungsverzeichnis dient.

Layoutabgleich
Der ↳KVV-Planer überarbeitet alle kommentierten ↳LV-Beschreibungen am PC, um ein einheitliches Aussehen zu gewährleisten.

Lehrplanbeauftragter
Der Lehrplanbeauftragte hat die Aufgabe der ↳Lehrveranstaltungsermittlung und ↳Deputatsverwaltung. Er beginnt seine Tätigkeiten ca. 1 Jahr vor Drucklegung.

Lehrtableau
Inhalt:
Veranstaltungsart, Titel, Semesterstundenzahl, Kern-/Vertiefungsgebiet, Teilnehmerzahl, Veranstalter
Umgang:
Zusammenführung aller ↳AB-Lehrtableaus zu einem großen Lehrtableau, das vom ↳FBR verabschiedet wird.

Lehrveranstalter
Personen, die befugt sind, Lehrveranstaltungen anzubieten. Vor allem Professoren, aber auch Dozenten und wissenschaftliche Mitarbeiter.

Lehrveranstaltungsermittlung
Der Lehrplanbeauftragte verschickt ↳AB-Lehrtableaus und ↳Deputatsformulare an die ↳AB-Lehrplanbeauftragten. Zu einem bestimmten Termin müssen diese Formulare wieder bei ihm eintreffen. Er überprüft die ↳AB-Lehrtableaus auf Korrektheit und führt sie zu einem ↳Lehrtableau zusammen, das er dann dem ↳FBR zur Verabschiedung zur Verfügung stellt.

Liste nicht fachbereichsgebundener Räume
Liste der vom Raumplaner angeforderten zentralen Vorlesungsräume, die von der Universitätsverwaltung vergeben werden.

LV-Beschreibung
Daten aus dem ↳Lehrtableau zusammen mit Raum-/Zeitvorschlägen der ↳AB-Lehrplanbeauftragten.

LV-Ermittlung
↳Lehrveranstaltungsermittlung

LV-Nummer
Die Lehrveranstaltungsnummern des Fachbereichs Informatik haben folgenden Aufbau und Bedeutung:

18.xxx	Fachbereich Informatik
18.0xx	Grundstudium
18.00x	Grundstudium, Vorlesung
18.01x	Grundstudium, Praktikum
18.03x	Grundstudium, Projekt
18.05x	Grundstudium, Proseminar
18.1xx	Hauptstudium, Vorlesung/Übung Kerngebiet
18.2xx	Hauptstudium, Vorlesung Vertiefungsgebiet
18.3xx	Hauptstudium, Projektseminar
18.4xx	Hauptstudium, Seminar/Oberseminar/Kolloquium
18.5xx	Veranstaltungen für andere Fachbereiche
18.6xx	Veranstaltungen anderer Fachbereiche für Informatiker
18.9xx	Tutorien

'x' steht dabei für eine beliebige Ziffer.
Um Veranstaltungen wie die Fachbereichsratssitzung bei der Raumvergabe berücksichtigen zu können, wurde der Schlüssel 90.xxx für sonstige Veranstaltungen eingeführt.

Pflichtveranstaltungen
Sowohl im Grundstudium als auch (eingeschränkt) im Hauptstudium gibt es Vorlesungen, die als Pflichtveranstaltungen bezeichnet werden. Diese Veranstaltungen müssen von allen Studenten besucht werden und dürfen daher nicht parallel angeboten werden.

Raumplaner
Der Raumplaner hat die Tätigkeiten der ↳Raumvergabe und der ↳VV-Erstellung durchzuführen.

Raumvergabe
Auf der Grundlage des vom ↳FBR verabschiedeten ↳Lehrtableaus erstellt der Raumplaner mit dem Stundenplanprogramm ↳Stupla ein ↳vorläufiges Raumverzeichnis. Das Entfernen etwaiger Überschneidungen wird im Gespräch mit den ↳AB-Lehrplanbeauftragten abgestimmt. Eine ↳Liste nicht fachbereichsgebundener Räume erhält der Raumplaner auf Anfrage nach Hörsälen von der Universitätsverwaltung.

Stupla
Stupla ist ein Stundenplanprogramm und unterstützt den ↳Raumplaner bei der ↳Raumvergabe, der Gewährleistung von Überschneidungsfreiheit bei ↳Pflichtveranstaltungen und bei der ↳VV-Erstellung.

Verlag
Ein Verlag übernimmt Druck und Vertrieb des ↳kommentierten Vorlesungsverzeichnisses.

Vorläufiges Raumverzeichnis
Wird vom ↳Raumplaner auf Grundlage des verabschiedeten ↳Lehrtableaus mit dem Stundenplanprogramm ↳Stupla erstellt. Nach ↳Absprachen mit den ↳AB-Lehrplanbeauftragten entsteht daraus das ↳VV.

Vorlesungsverzeichnis
Inhalt
LV-Nummer, Titel, Veranstalter, Zeit, Ort (siehe Anhang D)
Umgang:
Der ↳Raumplaner erstellt das Vorlesungsverzeichnis mit ↳Stupla, nachdem die ↳Raumvergabe abgeschlossen worden ist.

VV
↳Vorlesungsverzeichnis

VV-Erstellung
Nach Abschluß der ↳Raumvergabe erstellt der Raumplaner mit ↳Stupla ein Vorlesungsverzeichnis, das er an die ↳AB-Lehrplanbeauftragten und den ↳KVV-Planer verschickt. Außerdem erstellt er einen ↳VV-Text für das Universitäts-VV und sendet ihn an die Universitätsverwaltung.

VV-Text für das Universitäts-VV
Inhalt:
LV-Nummer, Titel, Veranstalter, Zeit, Ort
Umgang:
Auszug aus dem ↳VV, der im zentralen Universitätsvorlesungsverzeichnis abgedruckt wird.

Anhang B Glossar Berlin

Absprache
Die ↳ZUV kontaktiert einzelne Veranstalter telefonisch, wenn nach einem Ausweichtermin für einzelne Veranstaltungen gesucht wird. Zur offiziellen Bestätigung dieser neuen Termine werden ↳Korrekturfahnen versandt.

Erfassungsbeleg
Die ↳ZUV verschickt die Erfassungsbelege zur Anmeldung neuer Veranstaltungen.

Externe Firma
Die externe Firma hat eine reine Datenerfassungsfunktion. Sie erfaßt die Daten der ↳Erfassungs- und ↳Lektionsbögen und sendet die erfaßten Daten als ↳LV-Belege-Datenträger an die ↳ZUV zurück.

FB-Planer
Der FB-Planer übernimmt Korrektur- und Verteilungstätigkeiten. Er bekommt von der ↳ZUV die entsprechenden Formulare und leitet sie an die Veranstalter der jeweiligen ↳Institute weiter. Später sammelt er sie ein und sendet sie an die ↳ZUV zurück.

Hochschulbereiche
Äquivalent zu den einzelnen Fachbereichen der Universität Hamburg.

Institute der Fachbereiche
Jeder Fachbereich ist weiter in kleinere Organisationseinheiten unterteilt, die sogenannten Institute.

Korrekturfahnen
Die ↳ZUV erstellt die Korrekturfahnen und versendet sie zu den entsprechenden Veranstaltern, die mit der Bestätigung dieser Korrekturfahnen der Verlegung einer Veranstaltung zustimmen.

Lehrveranstaltungsbelege
Oberbegriff für ↳Erfassungsbelege und ↳Lektionsbögen.

Lektionsbogen
Der Lektionsbogen enthält die Daten von in einem bestimmten Turnus wiederkehrenden Veranstaltungen und wird vom Computer erstellt. Der Lektionsbogen muß vom entsprechenden Veranstalter bestätigt werden, bevor die Veranstaltung tatsächlich stattfindet.

LV-Belege-Datenträger
Die ↳externe Firma erfaßt sowohl die ↳Lektionsbögen als auch die ↳Erfassungsbögen elektronisch. Das Ergebnis dieser Arbeit ist der LV-Belege-Datenträger, der dann von der ↳ZUV in deren Rechner eingespielt wird und Grundlage der ↳Raum-Zeit-Planung ist.

Pflichtfach
Einige Veranstaltungen sind sogenannte Pflichtfächer, die von allen Studenten zu besuchen sind. Solche Veranstaltungen dürfen nicht parallel angeboten werden.

Raum-Zeit-Planung
Ein Systemdatenprogramm führt auf den im Computer zur Verfügung stehenden Daten eine erste Raum-Zeit-Planung durch. Dabei stehen als Informationen die von der ↳externen Firma erfaßten ↳LV-Belege-Datenträger zur Verfügung. Außerdem verwaltet das Programm alle Räumlichkeiten und kann anhand der Veranstaltungsnummer herausfinden, ob eine Veranstaltung ↳Pflichtfach, ↳Wahlpflichtfach oder eine andere Veranstaltung ist. ↳Pflichtfächer dürfen nicht parallel liegen. ↳Wahlpflichtfächer werden möglichst über-

schneidungsfrei gehalten. Sollte dies jedoch nicht gelingen, strebt die ↳ZUV eine persönliche ↳Absprache mit dem Veranstalter an, um eine Ausweichmöglichkeit zu suchen. Bestätigt wird die Verlegung mit der ↳Korrekturfahne.

Raumbelegungsplan
Ergebnis der ersten ↳Raum-Zeit-Planung ist der Raumbelegungsplan.

Schlüsselverzeichnisse
Die Schlüsselverzeichnisse dienen als Hilfsmittel für die Erstellung der ↳Erfassungsbögen. Mit ihrer Hilfe lassen sich ↳Hochschulbereich und ↳Zwischenüberschriften ermitteln, die den Hauptteil einer Veranstaltungsnummer ausmachen.

Stundenplanbeauftragter des Fachbereichs
↳FB-Planer

Systemdatenprogramm
Es handelt sich hierbei um ein Programm, das die Daten vom ↳LV-Belege-Datenträger genauer analysiert, die für die ↳Raum-Zeit-Planung relevant sind.

Textdatenprogramm
Es handelt sich hierbei um ein Programm, das die Daten vom ↳LV-Belege-Datenträger genauer analysiert, die für den Ausdruck im späteren ↳VV relevant sind.

Vorlesungsverzeichnis
Das Vorlesungsverzeichnis ist der nicht mehr änderbare, mit dem ↳Textdatenprogramm aufbereitete ↳Raumbelegungsplan.

VV
↳Vorlesungsverzeichnis

Wahlpflichtfach
Fächer, die von Studenten nach Neigungen belegt werden können. Eine gewisse Anzahl von Stunden muß aber erfüllt werden.

Zentrale Universitätsverwaltung
Die Zentrale Universitätsverwaltung ist für den gesamten Ablauf der Erstellung des ↳ Vorlesungsverzeichnis zuständig und verantwortlich.

ZUV
↳Zentrale Universitätsverwaltung

Zwischenüberschriften
Mit Hilfe der Zwischenüberschriften können Veranstaltungsnummern gebildet werden. Sie bestehen aus einem vierstelligen Schlüssel, über den sowohl Fachbereiche als auch eine dazugehörende Forschungseinheit mit der Unterteilung in Grund- und Hauptstudium eindeutig identifiziert werden kann.

Anhang C Vorstudie Berlin

Anhang D Vorlesungsverzeichnis Sommersemester 1994

Anhang E Anschreiben Druckformatvorlagen

Anhang F Deputatsformulare, Lehrtableau

Anhang G Literaturverzeichnis

Niels Erik Andersen: Professional Systems Development, New York u.a. 1990

Edward V. Berard: Object-Oriented Domain Analysis, in: Edward V. Berard: Essays on Object-Oriented Software Engineering, 1993

Kent Beck, Ward Cunningham: A laboratory for object-oriented thinking, SIGPLAN notices vol.24 no.10, 1989

Grady Booch: Object-Oriented Design with Applications, Benjamin/Cummings, Redwood City, CA 1991

Christiane Floyd et. al.: Anleitung zur konstruktiven Kritik, Einführung in die Softwaretechnik SS 1993, Universität Hamburg 1993

Christiane Floyd: Arbeitsunterlagen zur Lehrveranstaltung „Einführung in die Softwaretechnik“, Universität Hamburg, Ausgaben 1991 bis 1994

Niels Fricke, Christian Fürll, Holger Koschek, Thomas Pfohe, Peter v. Savigny: Systementwicklung - Projektmanagement, Projekt Softwaretechnik 1995

Erich Gamma: Objektorientierte Software-Entwicklung am Beispiel von ET++, Berlin u.a. 1992, Kap. 5.1.1

Stephan Herrmann: Objektorientierter Entwurf und Implementierung eines Systems zur Verzeichnisverwaltung anhand der Kriterien von PETS, Bericht Nr. 1994-11 der Technischen Universität Berlin, 1994

Klaus Kilberth, Guido Gryczan, Heinz Züllighoven: Objektorientierte Anwendungsentwicklung, Braunschweig, Wiesbaden 1994

Kimm, Koch, Simonsmeier, Tontsch: Enführung in Software Engineering, Berlin 1979

Bertrand Meyer: Objektorientierte Softwareentwicklung, München/Wien/London 1990

Jim Neighbors: Software Construction Using Components, University of California 1981

Bernd Page: Diskrete Simulation, Berlin u.a. 1991

Rubén Prieto-Díaz, Guillermo Arango: Domain Analysis Concepts and Research Directions, in: Rubén Prieto-Díaz, Guillermo Arango: Domain Analysis and Software Systems Modeling, IEEE 1991

Prieto-Díaz: Domain Analysis for Reusability, in: Will Tracz: Software Reuse: Emerging Technology, IEEE 1987

T. F. Thompson, W. J. Clancey: A Qualitative Modeling Shell for Process Diagnosis, IEEE Software 1986

Heinz Züllighoven: Arbeitsunterlagen zur Lehrveranstaltung „Objektorientierte Systementwicklung, Teil A: Entwurf“, Universität Hamburg 1994

Anhang H Abbildungsverzeichnis

Anhang I Stichwortverzeichnis