Studiengangsmodellierung

- Ein implementierter Diskussionsansatz -

Henning Brune[1], Marco Carolla[2], Thorsten Spitta[2]
Universität Bielefeld
[1] Dezernat IT/Orga, [2] Fakultät für Wirtschaftswissenschaften
{henning.brune | marco.carolla | thSpitta}@uni-bielefeld.de

Abstract: Dieses Papier skizziert das konzeptionelle Datenmodell einer Studiengangsmodellierung für Campus-Management-Systeme. Nach den ersten Pilotprojekten scheint dies ein Schlüsselproblem für die Standardisierung solcher Systeme zu sein. Das Modell beschränkt sich auf die Grunddaten von Studiengängen, wodurch prosaische Studienordnungen nicht berücksichtigt werden müssen. Die Darstellung erfolgt als Objektsicht mit Attributen und einer Beziehungssicht als UML Klassendiagramm. Vorhandene Implementierungen produktiver Systeme werden als Wissensbasis für die Validation des Modells benutzt.

1 Ein Grundproblem von Campus-Management-Systemen

Campus-Management-Systeme (CaMS) sind organisatorische Softwaresysteme für Hochschulen, die deren Verwaltung in den Gebieten Forschung und Lehre vereinfachen und transparent machen sollen [St+07, Br+09, De+09, Kl09, FH09, Ra09, SP+10]. Außer der „Baukasten-Sammlung“ des staatlichen Monopolisten HIS (Hochschul-Informations-Systeme) gibt es noch kein etabliertes Standardsystem, sondern nur eine Reihe von Pilot-Anwendungen in Universitäten und Fachhochschulen (s. [Sp97, Br07]).

Für nähere Darstellungen zu diesem Komplex sei auf die Literatur verwiesen, die viel Normatives und Prospektives bietet, aber noch nicht auf empirisch belastbare Erfahrungen verweisen kann (s. z. B. [BB10, BGS10]).

Ein Grundproblem solcher Systeme ist, dass sich in der Realität deutscher Hochschulen fast beliebige Strukturen von Regularien finden, die selbst zwischen Fakultäten stark inkompatibel sein können. Sie heißen *Prüfungsordnungen.* Es gibt einige übergeordnete Ansätze, wie Rahmenempfehlungen von bestimmten Berufsverbänden, deren Ziel es ist, halbwegs einheitliche Berufsbilder zu schaffen, z. B. die der Wirtschaftsinformatik, die einen großen Kanon von Inhalten vorschreibt oder empfiehlt [WK03]. Für die konkrete Abwicklung der Lehre in Hochschulen bedarf es aber rechtssicherer Prüfungsordnungen.

Üblicher Weise gibt es zu jeder Prüfungsordnung eine *Studienordnung*, die die Ausführungsbedingungen für die Studienabwicklung enthält. Da beide Dokumente Prosatexte mit Paragraphen als Kapitel oder Abschnitte sind, enthält eine Studienordnung regelmäßig hoch redundante Elemente zur Prüfungsordnung, da sie ja getrennt lesbar sein soll. Die z. B. von Hartmer beschriebene, traditionell übliche Struktur *vor* Bachelor/Master zeigt Bild **1**.

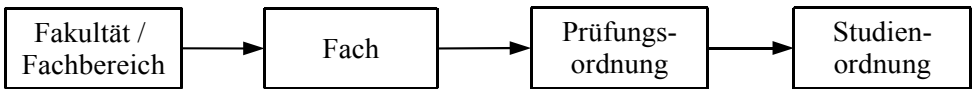

Bild 1: Traditionelle Struktur von Prüfungsvorgaben (skizziert nach [Ha11])

Es gibt also für jedes Fach eine Prüfungsordnung und zu jeder solchen, entsprechend ihrer zeitlichen Gültigkeit, mindestens eine Studienordnung. Das Problem einer solch prosaischen Wissensrepräsentation ist nicht nur, dass sie wenig „computergerecht“ ist. Dies alleine dürfte kein Grund für Änderungsbedarf sein, denn der Computer ist für den Menschen da und nicht umgekehrt. Das eigentliche Problem traditioneller Prüfungsordnungen ist der hohe Erstellungs- und Abstimmungsaufwand solcher Dokumente und das Prüfen der Zulässigkeit von erbrachten Leistungen durch die Prüfungsämter bei Abschlüssen. Insbesondere Änderungen von Prüfungsordnungen führen nach unseren Erfahrungen zu einem hohen Aufwand, die ordnungsgemäßen Studienverläufe zu organisieren und zu prüfen. Der letztgenannte Autor hat in 15 Dienstjahren diesen Aufwand der Änderung von Prüfungsordnungen zweimal geleistet und diese Arbeit in sehr schlechter Erinnerung.

Bei der Pilotanwendung des HIS Bausteins POS-GX für die Prüfungsverwaltung der wirtschaftswissenschaftlichen Fakultät der Universität Bielefeld wurde ein erster Ansatz zu einer Datenmodellierung des Diskursbereichs erarbeitet und der HIS GmbH überlassen [SM95, S. 147]. Dieses Datenmodell war zwar geeignet für studienbegleitendes Prüfen nach einem Kreditpunktesystem (vgl. auch [Sp+00]), aber nur im Rahmen einer starren Diplom-Prüfungsordnung. Die große Flexibilität, die BA/MA-Studiengänge aufweisen sollten, ließ sich auf Basis der einfachen Struktur von Bild **1** nicht bewältigen. Diese Flexibilität drückt sich z. B. darin aus, dass das Bachelorstudium zwar einen relativ festen Kern hat, die *Fachliche Basis*, aber in den Folgesemestern *Profile* aufweisen kann, die eine Vielzahl Spezialisierungen vorsehen können. Die entsprechenden Master-Studiengänge sind zum Teil auf den Profilen aufgebaut

Das Ziel einer Studiengangs-Modellierung war es [Br07, S. 9], die strukturellen Bestandteile aus Prüfungs- und Studienordnungen zu ermitteln, geeignete Abstraktionen zu schaffen und die weiterhin prosaischen Elemente der Prüfungsvorschriften zu typisieren. Hiermit sollten Abschlüsse mit sehr variablen Möglichkeiten in einem Datenbank-Schema abbildbar werden. Dies machte eine Redefinition üblicher, die Einführung neuer und die Abschaffung einiger traditioneller Begriffe notwendig. Bevor wir uns dem strukturellen Modell zuwenden, müssen die Begriffe geklärt werden, die später im Modell als Objekttypen auftauchen.

Der zuerst genannte Autor ist auch der maßgebliche Ideengeber und Realisierer des hier referierten Konzepts. Die Ideen zur Studiengangs-Modellierung gehen auf 2007 [Br07] zurück, die Realisierung erfolgte im Jahr 2012 als neues Release des Bielefelder Informationssystemens BIS, eines weit gehend vollständiges CaMS (s. [Br+09]).

2 Begriffe zur Modellierung

Wir notieren die wichtigsten Begriffe in einer erklärenden Aufzählung.

- Ein ***Studienmodell*** (SM) ist ein Exemplar von Regelungen für alle Studiengänge einer Hochschule, das für mehrere Jahre Gültigkeit hat. Bspw. gibt es in der Universität Bielefeld die SM 2002 und 2011 [UBi11].
- Ein ***Studiengang*** (Stg) ist in einer *Rahmenprüfungsordnung* festgelegt, die im Gegensatz zum traditionellen Modell für alle Fachdisziplinen gilt und zu einem Abschluss führt. Davon gibt es nur sehr wenige für eine Hochschule; typisch sind etwa die Ausprägungen {Bachelor of Arts, Bachelor of Science, Master of Science, Master of Arts und Doktor}. Letzteres gilt für einen sog. Promotionsstudiengang.
- Ein Studiengang führt zu einem ***Abschluss***, der durch ein amtliches Dokument bescheinigt wird, üblicher Weise *Zeugnis* genannt**.**

Jedes ***Fach*** eines Studiengangs wird verantwortet von einer ***Organisationseinheit*** (OE), üblicher Weise einer *Fakultät* bzw. einem *Fachbereich*. Für die Fächer gibt es in Ergänzung zu den Rahmenprüfungsordnungen – statt der früheren Prüfungs- und Studienordnungen – ***Fächerspezifische Bestimmungen*** (FSB), die das Curriculum vorsehen. Sie haben Bezüge zu den folgenden Strukturelementen, die die gewünschte Variabilität der Abschlüsse in maschinell prüfbarer Form gewährleisten. Wie im Folgekapitel zu sehen sein wird, sind die FSB das zentrale Element der Modellierung.

Zum einen gibt es ***Varianten*** (Var) als Kombination von Fächern. Sie stehen im Zusammenhang mit *Nebenfächern* und deren Beziehung zum *Hauptfach*. Es kann nur ein Hauptfach oder auch *Kernfach* erlaubt sein, aber auch Kombinationen aus Haupt- und Nebenfach. Ein Beispiel sind Lehramts-Studiengänge, die bestimmte Anteile von Haupt- und Nebenfach-Bestandteilen vorschreiben. Je nach angestrebtem Lehramt kann das „Hauptfach" eine andere Struktur haben.

Ein Studiengang kann unterteilt sein in ***Abschnitte***, die aufeinander aufbauen. Eine übliche Ausprägung sind vor allem beim Studiengang *Bachelor* die Abschnitte *Basis* und darauf aufbauend *Profil* (etwa Marketing, Controlling, Rechnungswesen in einem BWL-"Studiengang"). Denkbar ist aber auch die Variante mit genau *einem* Profil.

Das ***Modul*** ist auch in der teilweise aufgeheizten öffentlichen Diskussion um BA/MA ein zentraler Begriff. Es ist eine Lehreinheit, die z. B. Gegenstand von Prüfungen sein kann. Es ist in den FSB verankert und muss ***Elemente*** enthalten. Dies sind u. a. *Veranstaltungstypen* wie Vorlesungen, Seminare, Exkursionen oder Praktika, oft gekoppelt mit dem zweiten wichtigen Element, der *Leistung* (Klausur, Referat, Hausarbeit u. ä.). Bei beiden Elementtypen muss strikt zwischen dem abstrakten und dem konkreten Exemplar unterschieden werden. Ersteres gehört zu den *Grunddaten*, die relativ zeitlos sind, Letzteres zu den *Vorgangsdaten*, die immer einen Zeitbezug haben (zu Einzelheiten vgl. [SB08, Kap. 5]). Nur die Vorgangsdaten zu Veranstaltungen finden sich in einem Vorlesungsverzeichnis. Die Grunddaten sind in *Modulhandbüchern* niedergelegt und zertifizierungspflichtig.

Nicht alle Begriffe entsprechen im folgenden Abschnitt Objekttypen. So ist *Element* nur ein abstrakter Oberbegriff für die Grunddaten des konkreten Prüfungssgeschehens und *Abschnitt* ist nur ein Attribut als Aufzählungstyp. Auch die *Organisationseinheit* könnte als Aufzählungstyp implementiert werden, erscheint uns aber strukturell so bedeutsam, dass ein Objekttyp das Modell transparenter macht.

3 Datenmodell

Aus der knappen und sehr informellen semantischen Modellierung lässt sich ein grobes Datenmodell konstruieren. Die Darstellung erfolgt in zwei Formen, die beide für ein aussagefähiges Datenmodell erforderlich sind:

- **Beziehungssicht**, oft noch als Entity-Relationship-Modell, zunehmend auch als UML-Klassenmodell, das wir bevorzugen.
- **Objektsicht**, bei ER-Modellen sehr unübersichtlich, bei Klassenmodellen schlecht handhabbar. Wir benutzen aus pragmatischen Gründen eine relationale Darstellung, ohne in einen physischen Datenbankentwurf abgleiten zu wollen. Diese Art der Darstellung wird seit über 18 Jahren erfolgreich in der Lehre eingesetzt, weil die Prüfung der Fremdschlüssel-Integrität eine erste Validationsmöglichkeit ist, die in der Beziehungssicht völlig fehlt (vgl. im Einzelnen [SpBi08, Kap. 5 und 6]).

Wir beginnen mit der Objektsicht der Entitätstypen, da sie durch die Attribute am ehesten semantische Aussagen macht, was denn unter einem bestimmten Entitätstyp verstanden wird. Dies ist durch die „Erlangener Schule“ von Wedekind und Ortner seit Anfang der 80er Jahre bekannt [Ort85], wurde aber im Folgenden wenig beachtet (vgl. z. B. [Sch98]).

3.1 Objektsicht

Es folgen die relevanten Entitäts- (=Objekt)Typen in einer relationalen Darstellung. Namen werden auf pragmatische Weise so abgekürzt, dass überschaubare Zeichenketten und Lesbarkeit sich möglichst decken. Wir verwenden damit genau die Namen, die auch in den Datenbankschemata und den Variablennamen der Programme benutzt werden sollten.

Schlüssel sind mit '#' gekennzeichnet, Primärschlüssel sind zusätzlich unterstrichen, Alternativschlüssel sind mit '#a' gekennzeichnet. Ein Attribut mit '#' ohne Unterstreichung ist also ein Fremdschlüssel. Datentypen sind im Normalfall nicht genannt, zur besseren Verständlichkeit aber bei größeren Texten mit [tx] und bei Aufzählungstypen mit [en] (*enumeration*) ergänzt. Boolesche Variablen tragen ein '?' am Namensende.

3.1.1 Grunddaten

StudModell (Id#, Jahr#a, Name, gültig_von, gültig_bis)
Studiengang(StgNr#, Name, Kürzel)
OrgEinheit (Kürzel#, Name)

In Bild **2** wurde dieser Entitätstyp weg gelassen, da er sich als Aufzählungstyp modellieren lässt.

Fach (FachNr#, Name, Kürzel, OE.Kürzel#)
Abschluss (AbschlNr#, Name, Kürzel, SM.Jahr#, DiplSupplm [tx], DiplZugang [tx], Beschreibung [tx], Gruppe [en])
mit: Gruppe = {BA, MA}
Variante (VarNr#, Name, Kürzel, Erläuterung)
Var_Fach (VarNr#, FachNr#, StgNr#, Anteil)
Profil (Id#, Name, Kürzel, Abschnitt [en], FachNr#, StgNr#)
mit: Abschnitt = {Kern, Profil1, Profil2, -}
FächerSpezifische Bestimmung (Id#, ModName, ModNr#, Profil.Id#, AbschlNr#, StgNr#, FachNr#, VarNr#, Pflicht?, von.Sem#, bis.Sem#, empfohleneSemNr, Beschreibung [tx])
Student (Id#, MatrNr#a, Name, ImmatrikDat, Abschl#, StgNr#, FachNr#, VarNr#, Profil.Id#, Status [en])

Vom Studenten sind nur die für das von uns betrachtete Teilsystem relevanten Attribute genannt.

Modul (ModNr#, Name, Kürzel, Workload, OE.Kürzel#, Zyklus, Kompetenzen [tx], Inhalte [tx], Vorkenntnisse [tx], Vorbedingungen [tx], Beschreibung_weitere [tx], Verantw.Dozent#)

Man sieht, dass die stark textorientierte Modulbeschreibung durch bestimmte Felder strukturiert ist, ohne die Freizügigkeit einer Prosa-Beschreibung zu beschneiden.

Modul_Voraussetzung (ModNr#, VorModId#)

```
Modul_Struktur (ModNr#, Veranst.Id#)
Veranstaltung (Id#, Name, Art [en], Zyklus, Workload-Präsenz, Work-
  load-Home, Gruppengröße, Beschreibung [tx])
Leistung (Id#, Name, Kürzel, Dimension [en], Art [en])
             mit: Dimension = {Punkte, Stück, Tage, ...}
Modul_Leistg (Mod.Id#, Leist.Id#, Gewicht, Workload,
  Beschreibung [tx])
```

Diese Relation ist erforderlich für Leistungen ohne Bezug zu einer Veranstaltung, z. B. Praktika.

```
Veranst_Leist (Veranst.Id#, Leist.Id#, Gewicht, Workload,
  Beschreibung [tx])
```

3.1.2 Beispiele für Vorgangsdaten

Da wir uns hier auf die Studiengangs-Modellierung beschränken und nicht auch noch die Abwicklung der Lehre zum Thema haben, zeigen wir lediglich den Zusammenhang mit den Grunddaten an einem wichtigen Beispiel. Dies geschieht mittels der Objekttypen `Veranstaltung-VV` und der damit verbundene `Leistung-Expl`. Warum die Suffixe `'VV'` und `'Expl'`?

Jede konkrete Veranstaltung benötigt gegenüber dem „Template" der Prüfungsvorschriften natürlich Zeitbezüge. Der Zeitbezug ist prägendes Element aller Vorgangsdaten. Die Lehrveranstaltungen sind aber auch das „Bild", das die Studierenden für die konkrete Durchführung ihres Studiums haben, und für die Lehrenden ist es der Ort einer Computer unterstützten Kommunikation mit den Studierenden. Dieses ***Vorlesungs-Verzeichnis*** ist heute selbstverständlich im Inter- oder einem Intranet sichtbar und auch der Ort von Buchungen durch Studierende (z. B. Belegung von Veranstaltungen, Prüfungsanmeldungen oder Reservierung von Referatsthemen). Beim zweiten wichtigen Vorgang für die Durchführung von Lehre, der `Leistung`, ist zu viel Sichtbarkeit allein aus Datenschutzgründen nicht erwünscht; daher der Suffix ***Exemplar***. Bei einem Vorlesungsverzeichnis muss immer beachtet werden, dass neben einem geplanten und dem aktuellen Semester auch mehrere Jahre abgeschlossener Veranstaltungen für Studierende einsehbar sind.

Die Grunddatentypen **`Dozent`** und **`Raum`** (siehe Bild **2**) sind für ein Teilsystem *Lehre* essenziell, für das hier betrachtete Teilsystem Studiengänge jedoch nicht erforderlich.

Ein Datenmodell sollte aber immer auch Vorgangsdaten zeigen, denn dieser Teil der originären Daten ist die wichtigste Flussgröße aller Prozesse. Die Vorgangsdaten sind die „Spur", die organisatorische Prozesse hinterlassen können, in vielen Fällen auch müssen. Prüfungen müssen über lange Zeiträume rechtssicher sein, was nur mit Aufzeichnungen gelingen kann. Im Geschäftsleben schreibt das HGB (§§ 238, 257) die Aufzeichnung und Aufbewahrung der *Geschäftsvorfälle* vor.

Es folgen die Vorgangsdatentypen `Veranstaltung-VV` und `Leistung-Expl`. Der erste Vorgangsdatentyp erleichtert sicher die Abwicklung des Studiums, der zweite ist aufzeichnungs- und archivierungspflichtig. Dabei werden Redundanzen zu den Grunddaten in Abschnitt 3.1.1 vermieden. Bestimmte Attribute der Grunddaten können den Studierenden sichtbar gemacht werden, sollten aber in der konkreten Abwicklung nicht änderbar sein. Es muss aber möglich sein, in einem engen Rahmen bei der konkreten Veranstaltung von den Grunddaten abzuweichen. Wie hoch jeweils die Autonomie des Dozenten sein soll, muss die verantwortliche Organisationseinheit oder Institution regeln. Es werden ähnliche, aber nicht identische Namen für die Attribute der konkreten Veran-

staltung verwendet, um zu zeigen, dass es sich um Daten der Vorgänge handelt. Diese Attribute sind Eigenschaften des konkreten Vorgangs, die die Grunddaten nicht ändern können.

Veranstaltung-VV (Veranst.Id#, Sem#, VeranstVV-Id#a, Name, Art [en], Workl-Präs, Workl-Home, GruppGröße, Wochentag, Uhrzeit-von, Uhrzeit-bis, RaumNr#, Status [en], StartTermin, EndTermin, SWS, erwartete-Teilnehmer, TeilnehmerGrenze?, extDozent [tx], Beschreibung_Zusatz [tx], mitLeistung?)

Leistung-Expl (Leist.Id#, Sem#, LeistExpl-Id#a, MatrNr#, Art [en], VeranstVV-Id#, Datum, Note [en], Punkte, Text, OE.Id#, Anrechnung?, Prüfer.PersNr#, Beisitzer.PersNr#, ltztÄndDat, keineVeranst?)

Dazu der Grunddatentyp mit den hier relevanten Attributen:

Dozent (PersNr#, Name, Vorname, Titel-1 [en], Titel-2 [en], Status [en], ergänzenderText)

3.2 Beziehungssicht

Bild **2** zeigt die für die Studiengangs-Modellierung wesentlichen Entitätstypen als Klassenmodell. Bis auf wenige für das Problemverständnis wichtige Entitätstypen wie `Studienmodell` sind Aufzählungstypen im vorigen Abschnitt nur als Attribute modelliert, da dies die Struktur erheblich übersichtlicher macht. Ob man Aufzählungstypen als (sehr einfache) Datenbanktabellen implementiert, darf für ein konzeptionelles Modell keine Rolle spielen.

Der Kern einer Studiengangs-Modellierung besteht aus Grunddaten, die relativ zeitstabil sein sollten. Als Beispiel für deren Verbindung mit den Vorgangsdaten zeigen wir nur die Objekttypen Veranstaltung und Leistung.

Die Beziehungssicht der Objekttypen zeigt die zentrale Bedeutung der *Fächerspezifischen Bestimmungen* (FSB) für die Studiengangs-Modellierung, aber auch die Notwendigkeit, die zentralen Strukturelemente Modul, Veranstaltung und Leistung im Detail zu modellieren. In einer Implementierung wird FSB die zentrale Tabelle des Systems sein, die es je Studienmodell genau einmal gibt.

Ein Modell dieser Art hat eine gewisse Chance, den dynamischen Strukturexperimenten wenigstens mittelfristig zu folgen, die unsere bildungspolitische Landschaft seit einigen Jahren kennzeichnen. Nur mit flexiblen Strukturen können implementierte Campus-Management-Systeme auf Dauer nützlich *und* kostengünstig sein.

4 Zur Implementierung

Wir benutzen produktive Implementierungen als „Wissensbasis“, um unser konzeptionelles Modell zu validieren. Die Methodik und die drei bisher betrachteten Beispiele werden kurz skizziert.

4.1 Implementierungen als „Wissensbasis“

Schon die Objektsicht des konzeptionellen Modells zeigt, dass die Datenbasis eines CaMS alles Andere als einfach ist. Jede *implementierte* Basis wird noch erheblich komplexer sein. Dies macht es unwahrscheinlich, dass ein brauchbares Datenmodell wie im Lehrbuch top down entstehen kann. Man müsste viele Fachleute zu konzentrierter Workshop-Arbeit zusammen ziehen, was schwer organisierbar erscheint. Andererseits kann man allgemeingültige Datenmodelle nicht ohne empirisches Domänenwissen erstellen.

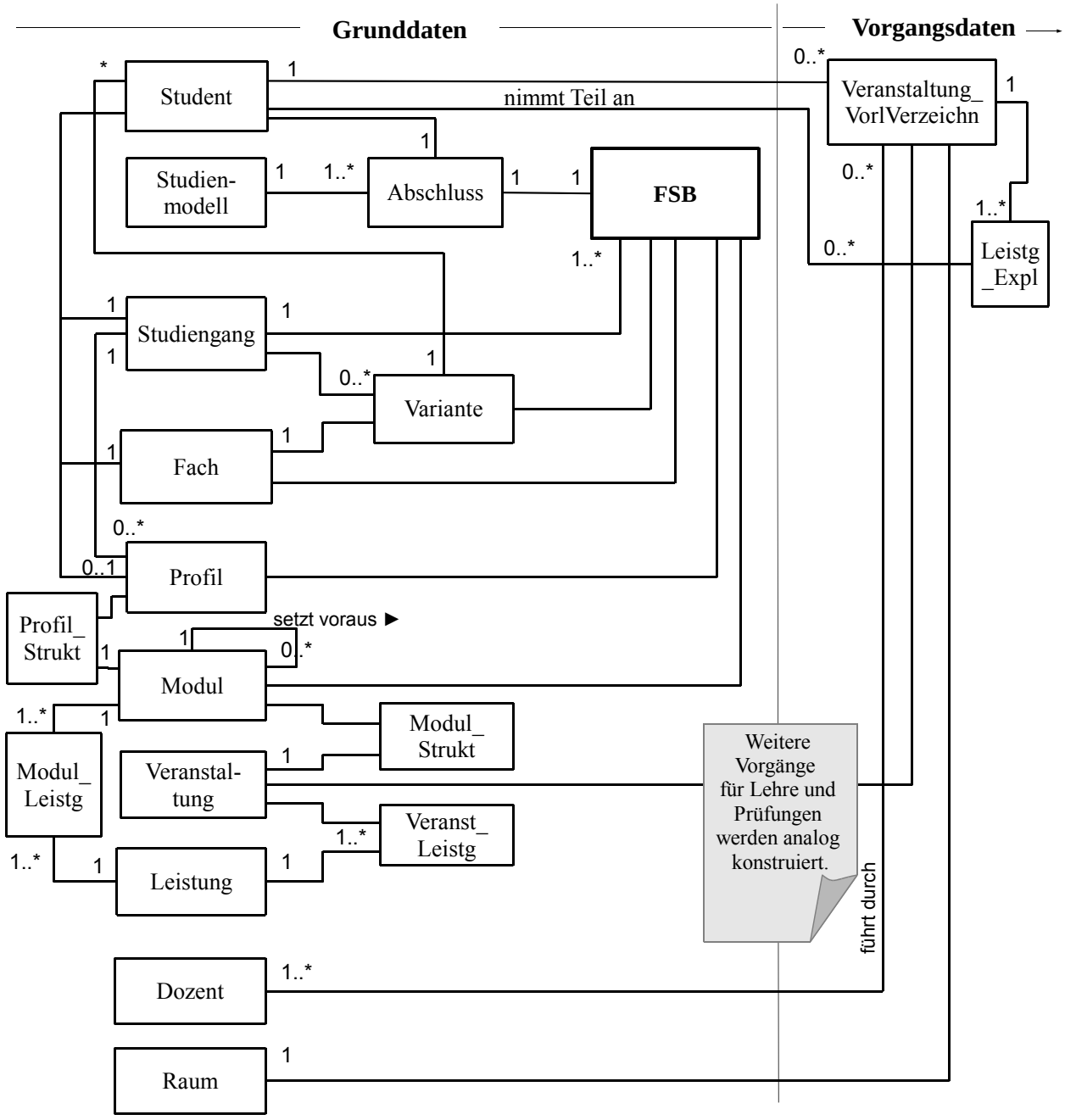

Bild 2: Grunddaten der Studiengangs-Modellierung mit einem Beispiel der Vorgangsdaten

Unser pragmatischer Ansatz geht bottom up vor. Die Autoren aus der Fakultät für Wirtschaftswissenschaften (Carolla & Spitta) analysieren im Rahmen eines Forschugnsprojektes die Datenbasen real eingesetzter Systeme und validieren daraus evolutionär das konzeptionelle Modell. Labormuster aus üblichen „Forschungsprojekten“ betrachten wir nicht. Wir fühlen uns zu dieser Haltung berechtigt, weil soziotechnische Softwaresysteme erst als „fertig“ gelten können, wenn sie sich in der sie benutzenden Organisation im Echtbetrieb befinden. Grechenig et al. weisen u. E. zu Recht auf die sehr hohe Benutzerkomplexität von Campus-Management-Systemen hin [Gr+10, 91].

Gegenwärtig stehen den genannten Autoren drei Implementierungen offen, von denen die ersten beiden dem hier dargestellten Modell zu Grunde liegen. Auch Implementierungen, die Standardsysteme werden wollen, etwa *Stine* von Datenlotsen, *SLCM* von SAP oder das Grazer System *Campus-Online* sollen noch hinzugezogen werden (s. auch [BGS10]).

4.2 Kurzhistorie dreier Implementierungen

Im Folgenden skizzieren wir die drei uns bekannten Implementierungen, wobei wir uns im Wesentlichen auf die Datenbasis beschränken.

4.2.1 HIS – Hochschul-Informations-Systeme

Der letztgenannte Autor fand zu seinen Dienstantritt als Hochschullehrer 1994 eine Installation von HIS POS vor (Prüfungsverwaltung), die er zusammen mit Jörn Mordau 1995 wie folgt beschrieb [SM95, 129]:

- „Unnötig komplexe Datenbasis infolge mangelhafter oder unterlassener Datenanalyse .... Im Fall HIS-POS fällt vor allem eine hohe Schlüsselredundanz auf, die das System undurchschaubar macht.
- Speicherung abgeleiteter Daten.
- Mangelhafte oder fehlende Integritätsbedingungen mit der Folge, daß die Datenbasis immer wieder inkonsistent wird. „Reparaturprogramme“, die direkt auf den Daten aufsetzen, erzeugen dann meist neue Inkonsistenzen.“

1996 stellte sich die Bielefelder Fakultät als Pilotanwender des neu entwickelten Release HIS POS-GX zur Verfügung. Statt des UNIX-Systems war vor die unveränderte Datenbasis (Informix auf einem NT-Server) ein Windows-Client gesetzt worden, der Anfragen an die Datenbank nach MS-Access in den Client importierte und damit eine deutlich flexiblere Benutzeroberfläche insbes. für Druckausgaben ermöglichte. Am Design der Datenbasis und weiterhin lückenhaften Integritätsbedingungen hatte sich nichts geändert.

Die „Datenbanken“ von HIS, auch die des Systems SOS, stammten offenbar aus dem Filesystem von Siemens-BS2000 Rechnern der 80er Jahre, die später nach UNIX portiert worden waren. Man sah den Tabellen an, dass sie offensichtlich für ein flaches Filesystem entwickelt worden und nie einem relationalen Entwurf unterzogen worden waren. Das Urteil über die Systeme und Datenbasen von HIS fiel in einem eingeladenen Vortrag der GI-Jahrestagung 1997 in Aachen entsprechend negativ aus [Sp97].

Das System POS-GX war – auslaufend für den Diplom-Studiengang – bis Ende 2011 in der Fakultät im Einsatz. Da wir im Laufe der 15 Jahre die Datenbasis kennen und in Teilen verstehen gelernt hatten, ließ sie sich für eine erste Validierung unseres hier vorgestellten Entwurfs verwenden.

4.2.2 BIS – Bielefelder Informationssystem

Das Bielefelder Informationssystem BIS wurde evolutionär seit 1998 entwickelt und ab 2002 mit den damals fertigen Funktionen und Datenbeständen universitätsweit eingesetzt (s. [Br+09]). 2007 wurde – auch in Zusammenarbeit mit der Fakultät für Wirtschaftswissenschaften – die Prüfungsverwaltung des BIS entwickelt und für die BA/MA Studiengänge in Betrieb genommen, während die Fakultät das HIS-System für den Diplom-Studiengang weiter betrieb.

Die Datenbasis wurde von vornherein für ein Zielsystem *Relationale Datenbank* entworfen, allerdings musste 2002 ein grundlegendes Reeingineering der Software-Architektur vorgenommen wer-

den, da auf ein proprietäres System gesetzt worden war (s. im einzelnen [Br+09, 489]): Ab dem neuen Release 2 mit einer eindeutigen Zugriffs-Schicht wurde Oracle als Datenbasis eingesetzt.

Die Dokumentation des Datenbasis-Entwurfs erfolgte mit pragmatischen Mitteln (heute Wiki-Dokumente). Diese Dokumentation wurde von Marco Carolla einem Redesign unterzogen, um sich in die Datenbasis semantisch einzuarbeiten. Dies ist die zweite „Wissensbasis" für unseren konzeptionellen Entwurf.

Die Datenbank-Implementierung enthält viele „historische" Attribute, da das System mit beiden Studienmodellen umgehen können muss, dem von 2002 und dem hier vorgestellten von 2011. Es befinden sich noch sehr viele Studierende „im System", deren Studium vor 2011 begonnen hat. Das Release „2.x" von BIS implementiert die vorgestellte Modellierung vollständig. Es befindet sich seit Oktober 2012 im Produktivbetrieb.

4.2.3 TISS – TU Wien Informations-Systeme und Services

Die TU Wien hat in ähnlicher Weise wie die U Bielefeld eine evolutionäre Entwicklung betrieben, allerdings mit wichtigen Unterschieden:

- Die TU Wien hat mit rund 37.000 doppelt so viele Studierende wie die U Bielefeld. Damit ist die User-Komplexität von TISS entsprechend größer als die von BIS, und zwar deutlich mehr als doppelt so hoch.
- Führend für die Entwicklung war ein Lehrstuhl für Softwaretechnik, der eng mit dem Leiter des Zentralen Informatikdienstes der Hochschule zusammen arbeitete, während in Bielefeld ein kleines Team, stark unterstützt durch den Leiter des zentralen Lehrerbildungs-Instituts, das Kernsystem entwickelte. Allerdings ist auch der Bielefelder Core-Entwickler Informatiker.
- Es mussten umfangreiche, in COBOL realisierte Altsysteme abgelöst werden. Dies bedeutete, dass eine umfassende Datenmigration zu leisten war. Teile der Altdaten waren zu übernehmen, andere überflüssig, wieder andere mit Fehlern behaftet. Wie stellt man fest, welche Daten zu welcher Kategorie gehören? (s. [St09])

Im November 2012 wurde auf der Tagung *Software-Management* über das Wiener System berichtet, das sich seit 2011 weitgehend vollständig im Echtbetrieb befindet [Gr+12].

Die Untersuchung der dortigen Datenbasis steht als Nächstes an.

5 Fazit

Eine Studiengangs-Modellierung erscheint möglich und nützlich. Erst diese dürfte für ein CaMS den Durchbruch für einen deutlich erkennbaren Nutzen für Hochschulen bringen. Erst damit wird nicht nur die Durchführung der Lehre im Kern, sondern auch die von Prüfungen und Abschlüssen vereinfacht und objektiviert.

Ohne Beachtung der Nutzenseite arten soziotechnische Systeme wie ein CaMS leicht in perfektionistische Kostenfallen aus, denn

einfach und „billig"
sind sie nicht zu haben.

Literatur

[BB10] Bode, A.; Borgeest, R. (Hrsg.): Informationsmanagement in Hochschulen**.** Springer, Berlin et al. 2010

[BGS10] Bick, M.; Grechenig, T.; Spitta, T.: Campus-Management-Systeme. In: *Pietsch, W.; Krams, B.* (Hrsg.): Vom Projekt zum Produkt. Fachtagung Dez. 2010, Lecture Notes in Informatics 178, Koellen, Bonn, 61-78.

[Br07] Brune, H.: Campus Management für Bielefeld – Eine aktuelle Perspektive. Arbeitspapier Universität Bielefeld, Okt. 2007.

[Br+09] Brune, H.; Jablonski, M.; Möhle, V.; Spitta, T.; Teßmer, M.: Ein Campus-Management-System als evolutionäre Entwicklung. In: [HKF09], 483-492.

[De09] Degenhardt, L.; Gilch, H.; Stender, B.; Wannemacher, K.: Campus-Management-Systeme erfolgreich einführen. In: [HKF09], 463-472.

[FH09] Fischer, H.; Hartau, C.: STiNE an der Universität Hamburg zur Einführung eines integrierten Campus Management Systems. In: [HKF09], 533-542.

[Gr+12] Grechenig, T.; Spitta, T.; Suppersberger, M.; Kleinert, W., Steininger, R.; Klar, C.; Pöll, M.: Entwicklung und Betrieb eines Campus-Management-Systems – Aspekte zur Nachhaltigkeit am Beispiel TISS. In: *Brandt-Pook, H.; Fleer, A.; Spitta, T.; Wattenberg, M.* (Hrsg.): Nachhaltiges Software Management, Nov. 2012, Lecture Notes in Informatics 209, Koellen, Bonn, 135-152.

[Ha11] Hartmer, M. (Hrsg.): Hochschulrecht. 2. Aufl., Müller, Heidelberg et al. 2011.

[HKF09] Hansen, H.R.; Karagiannis, D.; Fill, H-G. (Hrsg.): Business Services – Konzepte, Technologien, Anwendungen (Bd 2). 9. Int. Tagung Wirtschaftsinformatik, Febr. 2009 Wien.

[Kl09] Klug, H.: Erfolgsfaktoren bei der Umstellung von Informationssystemen in Hochschulen. In: [HKF09], 473-482.

[Ort85] Ortner, E.: Semantische Modellierung – Datenbankentwurf auf der Ebene der Benutzer. Informatik Spektrum 8(1985) 1, 20-28.

[Ra09] Radenbach, W.: Integriertes Campus Management durch Verknüpfung spezialisierter Standardsoftware. In: [HKF09], 503-512.

[SB08] Spitta, T., Bick, M.: Informationswirtschaft. 2.Aufl., Springer, Berlin et al. 2008.

[Sch98] Scheer, A.W.: ARIS – Modellierungsmethoden, Metamodelle, Anwendungen. 5.Aufl., Springer, Berlin et al. 1998.

[SG95] Schäfer, J. P.; Grauer, M. (Hrsg.): Universitätsverwaltung und Wirtschaftsinformatik. Proceedings, Siegen Okt. 1995.

[SK95] Sinz, E., Krumbiegel, J.: Gestaltung qualitätsgesicherter Universitätsprozesse am Beispiel des Prozesses ‘Lehre und Studium’. In: [SG95], 15-32.

[SM95] Spitta, T.; Mordau, J.: Entwicklung und Ergebnisse eines allgemeingültigen Fachkonzeptes für die Prüfungsverwaltung an Hochschulen. In: [SG95], 128-147. → HP

[Sp97] Spitta, T.: Standardsoftware zur Verwaltung und Führung von Fakultäten. GI-Jahrestagung `97, Univ. Bielefeld, Fakultät für Wirtschaftswissenschaften, Diskussionspapier Nr. 354, August 1997. → HP

[Sp+00] Spitta, T.; Decker, R.; Sigge, A.; Tiemann, V.; Wolf, P.: Erste Bilanz eines Kreditpunkte-Prüfungssystems. WiSt 29(2000) 10, 595-598. Langfassung: → HP

[Sp+10] Sprenger, J.; Klages, M.; Breitner, M. H.: Wirtschaftlichkeitsanalyse für die Auswahl, die Migration und den Betrieb eines Campus-Management-Systems. Wirtschaftsinformatik 52(2010) 4, 211-224.

[St09] Strobl, S. et.al.: Digging Deep: Software Reengineering supported by Database Reverse Engineering of a System with 30+ Years of Legacy. Wien, Vienna University of Technology (TU Wien), 2009.

[St+07] Stender, B.; Jablonski, M.; Brune. H.; Möhle, V.: Campus Management von der Hochschule aus gedacht, Wissenschaftsmanagement, (2007) 6, 19-26. s. auch: http://www.uni-bielefeld.de/bis/projekt/WiMa-Artikel, am 15.01.2013.

[UBi11] Universität Bielefeld: Erläuterungen zu den Rahmenprüfungsordnungen (2011). http://ekvv.uni-bielefeld.de/wiki/en/Erl%C3%A4uterungen_zu_den_%22Rahmenpr%C3%BCfungsordnungen%22_%28Studienmodell_2011%29#Einf%C3%BChrung, am: 16.01.2013.

[WK03] WKWI (Wissenschaftliche Kommission Wirtschaftsinformatik): Rahmenempfehlung für die Universitätsausbildung in Wirtschaftsinformatik. Informatik Spektrum 26(2003) 2, 108-113.

→ HP= http://www.wiwi.uni-bielefeld.de/im-ruhestand/wi-inf/publikationen.html