Stundenplan-Entwurfsunterstützung

Projekt Timetable

Helge Janicke, Niels-Peter de Witt, Karsten Wolke

8. Dezember 2001

Analyse und Entwurfsdokumente, Test

Dieses Dokument enthält alle für die Weiterentwicklung notwendigen Informationen und Teildokumente (Aufgabenbeschreibung, Machbarkeitsanalyse, Entwurf, Implementation und Test)

Falls Sie noch weiterführende Fragen zu den Dokumenten haben, wenden Sie sich bitte an eine der folgenden email-Adressen:

heljanic@agent.fho-emden.de
ndewitt@gmx.de
mail@karsten-wolke.de

oder besuchen Sie die Website

www.furchur.de

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Aufgabenbeschreibung

1.1 Allgemeine Beschreibung

Das Ziel der Stundenplan-Entwurfsunterstützung ist es, eine Applikation zu entwerfen, die das Erstellen von Stundenplänen erleichtert. Die Anwendung soll sich an dem Planungsverfahren des Fachbereichs Wirtschaft an der FHOOW orientieren.

1.1.1 Planungsverfahren im Fachbereich Wirtschaft

Das Ziel dieses Verfahrens ist es, einen möglichst optimalen Stundenplan für Professoren und Mitarbeiter zu erstellen. Dazu treffen sich die Lehrenden des Fachbereichs in einer mehrstündigen Sitzung. Es existiert eine Liste mit allen Lehrenden, sowie Stunden- und Raumpläne im DIN A2 Format, welche noch nicht ausgefüllt sind. Es ist ein Stundenplan für jedes Semester vorhanden. Der auf der Liste erstgenannte beginnt eine bestimmte Anzahl seiner Veranstaltungen in die Pläne einzutragen. Als nächstes kommt der zweitgenannte an die Reihe usw.. Der Letzte auf der Liste wählt zweimal und die Liste wird nun rückwärts wieder durchlaufen. Dieses Verfahren ermöglicht es den Lehrenden sich optimal abzusprechen (wegen Fahrgemeinschaften usw.). Jeder Lehrende hat Einfluß auf die Gestaltung des Stundenplanes. Während der Sitzung werden die eingetragenen Daten auf eine LOTUS Datenbank übertragen. Eventuelle Unstimmigkeiten, sofern sie nicht schon innerhalb der Sitzung erkannt werden, werden im Nachhinein von einer Sekretärin geklärt.

1.1.2 Ziel des Projektes

Da im Fachbereich Wirtschaft bereits eine gut organisierte Datenbank existiert und im Fachbereich E&I kein Intresse an einer neuen Entwurfsmethode besteht, konzentriert sich dieses Projekt auf die Visualisierung der Daten während der oben beschriebenen Sitzung. Die Idee ist, die Stunden und Raumpläne in einer geeigneten grafischen Oberfläche darzustellen. Die so eingegebenen Daten sollen direkt in eine Datenbank gespeichert werden, mit deren Hilfe Unstimmigkeiten erkannt werden oder gar nicht erst entstehen können.
Eine konkrete Anbindung an eine bestehende Datenbank steht nicht im Vordergrund des Projekts, soll aber möglichst einfach zu realisieren sein.

2 Machbarkeitsanalyse

2.1 Funktionale Anforderungen

In diesem Abschnitt werden die Anforderungen, die an das Projekt gestellt werden benannt. Es wird hier in die Bereiche Konzepte, Prototypen, Module und Dokumentation unterschieden.

2.1.1 Konzepte

Konzepte sind die den Prototypen und Modulen zu Grunde liegenden Überlegungen. Anhand dieser Konzepte wird das Produkt fertiggestellt.

K_GUI Entwurf einer Grafischen Oberfläche.
Wie soll die grafische Oberfläche gestaltet werden. Ergonomie, Intuitive Steuerung sind die Hauptgesichtspunkte des Auftraggebers.

K_DB_Java Entwurf einer Datenbankschnittstelle.
Die Daten sollen in einer Datenbank gehalten werden. Die Ansteuerung der Datenbank soll möglichst gut portierbar auf verschiedene Datenbanksystem sein.

K_DB Modellierung einer Datenbank.
Für die Datenbank wird ein ER-Modell[1]erstellt. Ein vollständiges Data Dictionary (DD) bildet die Grundlage für das Generieren der Datenbank.

2.1.2 Prototypen

Prototypen dienen ausschließlich der Rücksprache mit dem Auftraggeber, sowie zu Versuchszwecken während der Entwurfsphase. Sie beinhalten keine vollständige Funktionalität und sind nicht frei von Mängeln.

P_GUI Prototyp einer grafischen Oberfläche.
Es ist ein Prototyp der grafischen Oberfläche zu erstellen. Der Prototyp soll die Bedienung des Programms veranschaulichen. Es sind keine Funktionalitäten zu implementieren.

P_DB_Java Prototyp der Datenbankschnittstelle. Die Anbindung an die Datenbank soll an einem Prinzipbeispiel gezeigt werden.

P_SimDB Simulieren einer Datenbank. Für den Prototyp der grafischen Oberfläche sollen Beispieldaten bereitgestellt werden. Die Datenbankschnittstelle soll Testdaten zurückliefern, ohne auf die Datenbank zuzugreifen.

2.1.3 Module

Module sind die endgültig eingesetzten Teile des Projektes. Sie müssen so konstruiert sein, daß sie sich bei Bedarf leicht ersetzen lassen.

M_DB Datenbank aus K_DB generieren.
Aus dem Konzept (ER-Modell) soll die Datenbank erstellt werden. Dieses geschieht mit einem frei gewählten Datenbanksystem.

M_Daten Beispieldaten in Datenbank M_DB erzeugen.
Die Datenbank ist mit Beispieldaten zu füllen, so daß es möglich ist mit den Daten die verschiedenen SQL-Ausdrücke der Abfragelogik zu testen.

M_DB_Java Entgültige Fassung der Datenbankschnittstelle.
Diese endgültige Fassung soll alle benötigten SQL-Ausdrücke enthalten, die für das Programm notwendig sind, sowie die Anbindung an die Datenbank.

[1]Entity-Relationship-Modell, ein von Peter Chen entwickeltes Schema zur Datenmodellierung

M_GUI Endgültige Fassung der GUI[2].
Ist die vollfunktionsfähige grafische Oberfläche, über die sich das Programm steuern lässt.

2.1.4 Dokumentation

Dokumentationen werden im PDF Format erstellt. Bei der Erstellung orientieren wir uns am Prozesshandbuch des SWT-Praktikums. Handschriftliche Entwürfe sind der Original-Dokumentation beigelegt. Quelltexte werden direkt *in line* dokumentiert, dies geschieht mit JAVADOC. Die daraus automatisch erzeugte HTML Dokumentation ist in digitaler Form erhältlich.

- Dokumentation der Entwürfe:

  D_Entw_GUI grafische Oberfläche

  D_Entw_DB_Java Datenbankschnittstelle.

  D_Entw_DB Datenbankmodell (ERM, DD).

- Dokumentation der Prototypen:

  D_Prot_GUI grafische Oberfläche

  D_Prot_DB_Java Datenbankschnittstelle.

- Dokumentation der Module:

  D_Mod_DB Datenbank

  D_Mod_DB_Java Datenbankanbindung und Abfragelogik

  D_Mod_GUI grafische Oberfläche.

2.2 Randbedingungen

Implementierungssprache JAVA

Eine gute Bedienbarkeit soll den Planungsprozeß beschleunigen.

2.3 Qualitätsmerkmale

Die Tests, die für die Einhaltung der einzelnen Qualitätsmerkmale durchgeführt werden, sind hier nur kurz angeschnitten. Eine ausführliche Beschreibung finden sie in dem Kapitel Tests. Dort wird die Vorbereitung, Durchführung und Auswertung der verschiedenen Tests behandelt.

Q_Bedie Bedienbarkeit
Unter Bedienbarkeit wird die anwenderfreundliche Bedienung
verstanden. Dieser Punkt bezieht sich also auf die Grafische Oberfläche. Wir unterscheiden drei Maße:

Clicks Die Anzahl der Mouseklicks für die Zuordung von Termin, Fach, Raum und Lehrkraft.

Schachtelung Die Schachtelungs-Tiefe der Menüs.

Note Bewertung durch unabhängige Testpersonen. Die Note 6 ist schlecht, 1 entspricht sehr gut.

[2] Graphical User Interface

Bedienbarkeit – Quantifizierung			
Maß	Muß	Soll	Wunsch
Clicks Schachtelung Note	8 4 4	4 2 2	1 1 1
Test			
Clicks	Durchschnittswert aller Zuordungen während einer Stundenplanerstellung.		
Schachtelung	Zählen der Schachtelungstiefe.Bewertet wird anhand der tiefsten Schachtelung.		
Note	Mittelwert der vergebenen Noten.		

Q_Erwei Erweiterbarkeit

Unter Erweiterbarkeit verstehen wir, die Anzahl der, für die Erweiterung, zu ändernden Zeilen im bereits vorhandenen Quellcode. Gemessen wird in LOC.

Erweiterbarkeit – Quantifizierung			
Maß	Muß	Soll	Wunsch
LOC	150	40	0
Test			
LOC	Bei einer Erweiterung werden die im vorhandenen Code geänderten Zeilen gezählt.		

Q_Zuver Zuverlässigkeit

Unter dieses Qualitätsmerkmal fällt die Ausfallhäufigkeit des Programms. Gemessen wird in Abstürzen pro Bedienstunde (Crash/ h).

Zuverlässigkeit – Quantifizierung			
Maß	Muß	Soll	Wunsch
Crash/ h	0.2	0.02	0
Test			
Crash/ h	Der Test zur Bedienbarkeit (Q_Bedie) kann gleichzeitig dazu verwendet werden die Absturzrate zu messen. Voraussetzungen für diesen Test sind die gleichen wie bei Q_Bedie. Es wird der Durchschnitt über die aufsummierte Bedienzeit aller Benutzer gebildet.		

Q_Porti Portierbarkeit

Gemessen werden die Anzahl der geänderten Zeilen (LOC) für eine Anpassung an ein anderes System bzw. eine andere Datenbank.

Portierbarkeit – Quantifizierung			
Maß	Muß	Soll	Wunsch
LOC	150	50	0
Test			
LOC	Bei einer Portierung werden die im vorhandenen Code geänderten Zeilen gezählt.		

Q_Konsi Konsistenz

Gemessen werden die Inkonsistenzen pro Datenmanipulation.

Konsistenz – Quantifizierung			
Maß	Muß	Soll	Wunsch
s.o.	0	0	0
Test			
	Getestet wird die Anzahl der entdeckten Inkonsistenzen während der Nutzung in anderen Testphasen.		

Q_Robus Robustheit
Unter Robustheit werden die nicht erkannten Eingabefehler pro Bedienstunde verstanden (Faults/ h).

Robustheit – Quantifizierung			
Maß	Muß	Soll	Wunsch
Faults/ h	0.2	0.02	0
Test			
Faults / h	Bei dem Test der Bedienbarkeit werden nicht abgefangene Fehleingaben protokolliert. Einen alternativen Test stellt ein Eingabe-Roboter dar, bei dieser Testmethode muß allerdings in Faults/ Input gerechnet werden, da ein Roboter sehr viele Eingaben in einer Stunde simulieren kann.		

Q_Reakt Reaktionszeit
Dauer bis das Ergebnis einer Anfrage für den Bediener sichtbar wird. Die Reaktionszeit wird in Sekunden gemessen.

Reaktionszeit – Quantifizierung			
Maß	Muß	Soll	Wunsch
sec	5	1	0.1
Test			
sec	Da der größte Rechenaufwand bei der Datenbankabfrage liegt, wird diese Zeit gemessen. Falls trotz angemessener Geschwindigkeit der Datenbankabfrage die Reaktionszeit als langsam erscheint (Benutzertest) werden weitere Tests notwendig.		

Q_Erler Erlernbarkeit
Die Erlernbarkeit wird von den Bedienern benotet.
6 entspricht schlecht, 1 sehr gut.

Erlernbarkeit – Quantifizierung			
Maß	Muß	Soll	Wunsch
Note	4	2	1
Test			
Note	Mittelwert der vergebenen Noten.		

Q_Wartb Wartbarkeit
Die Wartbarkeit drückt den Aufwand aus, der zur Behebung eines Fehlers benötigt wird. Gemessen wird der Aufwand in Minuten pro Fehler.

Wartbarkeit – Quantifizierung			
Maß	Muß	Soll	Wunsch
Minuten	600	60	10
Test			
Minuten	Treten während der Testphase Fehler auf wird die Arbeitszeit gemessen, die zur Behebung des Fehlers notwendig ist.		

2.4 Lösungen

L_Modul Trennen von GUI, Logikeinheit, und Datenhaltung.
Die Modularisierung von Programmen ist eine Maßnahme, die der guten Erweiterbarkeit Rechnung tragen soll. Einzelne Module lassen sich auch besser warten.

L_StdSQL Verwendung von Standard-SQL (SQL92).
Die Verwendung von Standard SQL soll eine gute Portierbarkeit auf andere Datenbanksysteme (DBS) ermöglichen. Leider erfüllen noch nicht alle DBS diesen Standard.

L_PropSQL Verwendung von proprietären SQL.
Herstellerspeziefische Abfragemöglichkeiten sind meistens performanter, als die durch Standards beschriebenen Funktionen. Wenn hochperformante Abfragen benötigt werden ist dies eine gute Lösung.

L_Proxy Puffern von schon gelesenen Daten.
Das Zwischenspeichern von bereits gelesenen Daten im Hauptspeicher erspart zeitaufwändige IO-Zugriffe und beschleunigt Zugriffszeiten.

L_Intuitiv Eine intuitive Bedienung wird durch eine ansprechende Oberfläche und die Verwendung von Standardkomponenten (Listen, Drop-Down-Menüs) sowie eine ansprechende Steuerung (Drag&Drop) erreicht.

L_Help Allgemeine Hilfe-Dokumente.
Ausreichend viele Hilfedokumente ermöglichen es interessierten Benutzern, sich tief in die Möglichkeiten eines Systems einzuarbeiten.

L_Statusbar Anzeigen von aktuellen Hilfen/ Erläuterungen.
Ist für nicht geübte Benutzer von großer Hilfe, da auch bei Kenntnis der Funktionsweise noch eine stichpunktartige Hilfe gegeben werden kann.

L_EingBesch Beschränken der Eingabemöglichkeiten.
Die Einschränkung der Eingabemöglichkeiten verhindert Falscheingaben. Wenn dem Benutzer nur eine eingeschränkte Auswahl zur Verfügung steht kann er sich schneller entscheiden. Die Auswahlmöglichkeiten dürfen nicht zu restriktiv sein, also alle sinnvollen Eingaben gestatten.

L_Test Testen der einzelnen Module/ Funktionalitäten.
Gute Tests finden viele Fehler, die dann vor der Auslieferung an den Anwender noch beseitigt werden können. Tests stellen auch eine Sicherheit dar, wenn später Fehler auftreten und Regressansprüche geltend gemacht werden.

L_Java Verwendung von Java als Programmiersprache.
Die Verwendung von Java ist von Vorteil, da es eine freie Programmierspache ist. Die Objektorientiertheit ermöglicht es, Modelle so umzusetzen, daß sie im Code noch erkennbar sind. Die Standards und Eigenschaften von Java unterstützen eine weitestgehend fehlerfreie Programmierung.

L_DB Sicherung der Datenkonsistenz (Relationen, Primärschlüssel).
Über Relationen und die Primärschlüssel der Datenbank wird erreicht, daß viele Inkonsistenzen in den persistent gespeicherten Daten nicht entstehen können. Relationen haben eine ausgewiesene Multiplizität, die nicht ausversehen überschritten werden kann (z.B. Jeder Mensch hat nur eine leibliche Mutter Multiplizität n:1). Der Primärschlüssel muß innerhalb einer Relation immer eindeutig sein. (z.B Es gibt maximal ein Buch mit der ISBN Nummer xxxxxx in der Liste aller Bücher).

L_Normalisier Normalisierung des ERM.
Die Normalisierung dient dazu, Redundanzen durch Abhängigkeiten von Attributen zu verhindern.

L_Exklusiv Exklusive Nutzung der DB.
Wenn das Programm in einem verteilten Umfeld (z.B. Internet) genutzt werden soll, kann über eine exklusive Datenbanknutzung das Problem asynchroner Zugriffe leicht gelöst werden.

L_Nativ Systemabhängiger (nativer) Code.
Falls performantere Funktionen gewünscht werden; ist es möglich den Java-Bytecode in Maschinencode umzuwandeln, hiermit lassen sich Ausführungsgeschwindigkeiten erreichen, die fast an C-Programme herankommen. Eine andere Möglichkeit ist die Nutzung von C-Programmen, auf die über die JNI-Architektur zugegriffen wird.

L_UserGuide How to start...
Eine kurze Einführung in die Grundfunktionen des Programms erleichtern den Start.

L_SunSTD Verwendung des SUN-Programmier-Standards.
Erhöhen die Lesbarkeit des Quelltextes. Damit wird die Fehlersuche leichter, und die Einarbeitungszeit Anderer kürzer.

L_ProgGuide Javadoc Dokumente.
Bilden den Quasi-Dokumentationsstandard für Java-Programme. Die Nutzung dieser Dokumentation erleichtert die Einarbeitung in das Programm zur Weiterentwicklung.

L_AnaDoc Analyse Dokumente
Gut dokumentierte Entwürfe zeigen das Konzept, das hinter dem Programm steht. Verschiedene Diagramme (Ablauf-, Sequenz-, Klassendiagramme etc.) lassen sich leichter lesen als Quelltexte.

L_Log Logfile Generierung bei Fehlern.
Wenn im Fehlerfall ein Logfile erstellt wird, mit Informationen, die zur Fehlerfindung relevant erscheinen, ist es möglich, den Fehler in kürzerer Zeit zu isolieren und zu beheben.

2.5 Bewertung der Lösungen

Die im vorangegangenen Abschnitt aufgelisteten Lösungen werden hier den Qualitätsmerkmalen gegenübergestellt. In der entstandenen Matrix läßt sich gut abschätzen, welche Lösungen mehr Vorteile, bzw. Nachteile, bringen als andere.

Tabelle 1: Gegenüberstellung der Lösungen und Qualitätsmerkmale

	Q_Bedie	Q_Erwei	Q_Zuver	Q_Porti	Q_Konsi	Q_Robus	Q_Reakt	Q_Erler	Q_Wartb
L_Modul	0	++	+	0	0	0	-	0	++
L_StdSQL	0	0	0	++	0	0	-	0	+
L_PropSQL	0	0	0	--	0	0	+	0	-
L_Proxy	0	0	0	0	--	0	+	0	0
L_Intuitiv	++	0	0	0	0	+	0	++	0
L_Help	++	0	0	0	0	+	0	++	0
L_Statusbar	++	0	0	0	0	+	0	++	0
L_EingBesch	-	0	+	0	++	++	0	0	0
L_Test	0	0	++	0	0	++	0	0	+
L_Java	0	++	0	++	0	0	-	0	+
L_DB	0	0	0	0	++	0	0	0	+
L_Normalisier	0	0	0	0	++	0	0	0	-
L_Exklusiv	0	0	0	0	++	0	-	0	+
L_Nativ	0	-	0	--	0	0	++	0	-
L_UserGuide	++	0	0	0	0	+	0	++	0
L_SunSTD	0	++	0	0	0	0	0	0	++
L_ProgGuide	0	++	0	+	0	0	0	0	++
L_AnaDoc	0	++	0	+	0	0	0	0	++
L_Log	0	0	0	0	0	0	0	0	++

LEGENDE: ++ ...positiver Einfluß auf Qualitätsmerkmal
-- ...negativer Einfluß auf Qualitätsmerkmal

2.5.1 Nicht näher betrachtete Lösungen

L_PropSQL Wir verzichten soweit es möglich ist auf proprietäre SQL-Anweisungen, um die Nutzung diverser Datenbanken zu ermöglichen. Falls es sich, aus Performancegründen, als notwendig erweist den Datenbankzugriff zu optimieren, kann aber durchaus auf die gute Portierbarkeit verzichtet werden.

L_Proxy Ein Proxy stellt einen höheren Programmieraufwand dar, der eventuell nicht notwendig ist. Sollte es jedoch Performance-Probleme geben kann es sich als günstig erweisen bereits gelesene Daten zwischenzuspeichern.

L_Nativ Auch hier gilt das gleiche wie bei den vorangegangenen Punkten. Bei großen Performance-Problemen kann auf nativen Code zurückgegriffen werden.

L_Log Auf das Erstellen von Log Files wird in der Anfangsphase verzichtet. Vielleicht werden die Log Files aber im Zuge des Tests implementiert.

L_Exklusiv Die Exklusive Nutzung der Datenbank ist in der Anfangsphase nicht von Interesse, da so oder so nur eine Anwendung gleichzeitig auf die Datenbank zugreift. Wird das Programm auf mehrere Rechner verteilt, gilt es die Frage der exklusiven Nutzung neu zu stellen. Die meisten Datenbanken, bzw. Treiber bieten schon die Möglichkeit der Benutzereinschränkung.

2.6 Risiken bei der Erfüllung der Qualitätsmerkmale

Im folgendem werden die Risiken abgeschätzt, die der Erfüllung eines Qualitätsmerkmals im Wege stehen könnten.

rskPerformance schlechte Performance.
Betroffene Qualitätsmerkmale:

- Q_Bedien
- Q_Reakt

Lösungen die Qualitätsmerkmale sichern:

- L_PropSQL
- L_Proxy

- L_Nativ
- L_Exclusiv

Da sich in der Datenbank nur sehr wenige Datensätze befinden, und die Abfragen zumeist auf einer lokalen Kopie stattfinden, sehen wir in der Performance kein großes Risiko.

rskDatenbank Probleme bei der Datenbankanbindung/ -erstellung.
Betroffene Qualitätsmerkmale:

- Q_Konsi
- Q_Porti
- Q_Wartb

Lösungen die Qualitätsmerkmale sichern:

- L_StdSQL

Sehen wir auf Grund unserer Erfahrungen als kein schwerwiegendes Problem an. Falls sich doch unerwartete Probleme zeigen sollten, besteht die Möglichkeit Fachleute um Rat zu fragen.

rskDBPort Anpassen an andere Datenbanken.
Betroffene Qualitätsmerkmale:

- Q_Porti

Lösungen die Qualitätsmerkmale sichern:

- L_StdSQL
- L_AnaDoc

Es kann zu Problemen führen, wenn aus Performancegründen kein Standard SQL verwendet wurde, oder die Ziel-Datenbank diesen Standard nicht unterstützt.

rskSWQuali Mangelhafte Qualität der Software.
Betroffene Qualitätsmerkmale:

- Q_Erwei
- Q_Zuver
- Q_Porti
- Q_Robus
- Q_Wartb

Lösungen die Qualitätsmerkmale sichern:

- L_AnaDoc
- L_Modul
- L_Test
- L_Java
- L_SunStd
- L_JavaDoc
- L_Log

Durch Qualitätsicherungsmaßnahmen wird die Software ständig in ihrer Qualität geprüft und überwacht. Eine von uns durchgeführte QS ist das *Über die Schulter schauen* (X-Treme Programming).

Qualitätsmerkmale die hier nicht aufgeführt sind beinhalten, unserer Meinung nach, keine nennenswerten Risiken.

3 Analyse

3.1 Anwendungsfälle

In diesem Abschnitt werden die denkbaren Anwendungsfälle näher betrachtet. Die Anwendungsfälle werden in Abbildung 1 grob dargestellt. In weiteren Abbildungen werden Anwendungsfälle noch weiter unterteilt. Die Anwender (blaue Kasten außen) stellen keine Personen sondern Rollen dar. Ein Informationssuchender kann also auch gleichzeitig ein Lehrender sein.

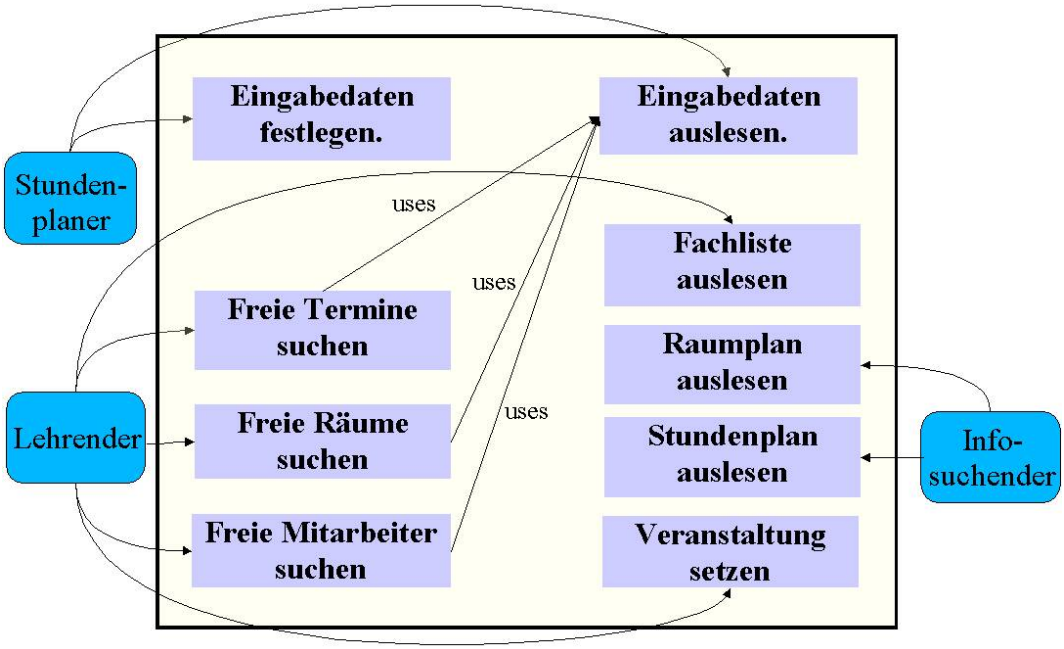

Abbildung 1: Hauptanwendungsfälle [Anwendungsfall01.jpg]

3.1.1 Beschreibung der Hauptanwendungsfälle

Die in obiger Grafik dargestellten Anwendungsfälle stellen
keine endgültigen Anwendungsfälle dar, sondern sind vielmehr eine Zusammenfassung von spezielleren Anwendungsfällen.

Eingabedaten festlegen Unter Eingabedaten verstehen wir alle Eingaben, die im Vorfeld notwendig sind um einen Stundenplan zu erstellen. Eingabedaten für die Erstellung sind:

- Raumliste
- Mitarbeiterliste
- Fachliste
- Termine
- Semester (Klassen)

Eingabedaten auslesen Eingabedaten müssen zur Kontrolle auch wieder ausgelesen werden können.

Freie Termine suchen Freie Termine werden meist für einen Mitarbeiter und ein Fach gesucht. Es muß eine Konsistenzprüfung stattfinden. Konsisitenzprüfungen sind z.B. Ob der Mitarbeiter zu diesem Zeitpunkt schon einen anderen Termin wahrnimmt, ob die zuhörenden Semester an diesem Termin eine andere Vorlesung haben etc.

Freie Räume suchen Zu einem Termin und einem Fach muß eine Liste mit allen verfügbaren Räumen geliefert werden. Der Raum muß die Kriterien (z.B Ausstattung) erfüllen, die für das Fach gefordert werden.

Freie Mitarbeiter suchen Die Möglichkeit aus allen Mitarbeitern diejenigen heraussuchen, die zu einem Termin assistieren können.

Fachliste auslesen Die Liste mit allen Fächern auslesen, die ein Lehrender noch zu verteilen hat.

Raumplan auslesen, Stundenplan auslesen Den aktuellen Stand der Stundenpläne und Raumpläne ausgeben.

Veranstaltung setzen Eine Veranstaltung ist eine Vorlesung, die zu einem Fach stattfindet. Eine Vorlesung setzt sich aus Informationen über Fach, Hörerschaft, Raum und Termin zusammen.

3.2 Zustandsdiagramme

Die folgenden Zustandsdiagramme stellen zum einen den *allgemeinen Programmablauf* (Abbildung 2), zum anderen den Ablauf beim *Festlegen einer Veranstaltung* (Abbildung 3), dar. Die Zustandsdiagramme für die Anwendungsfälle *Raumplan anzeigen*, bzw. *Semester-Stundenplan anzeigen* (Abbildung 4) sind zwar sehr einfach, werden der Vollständigkeit halber hier trotzdem abgedruckt.

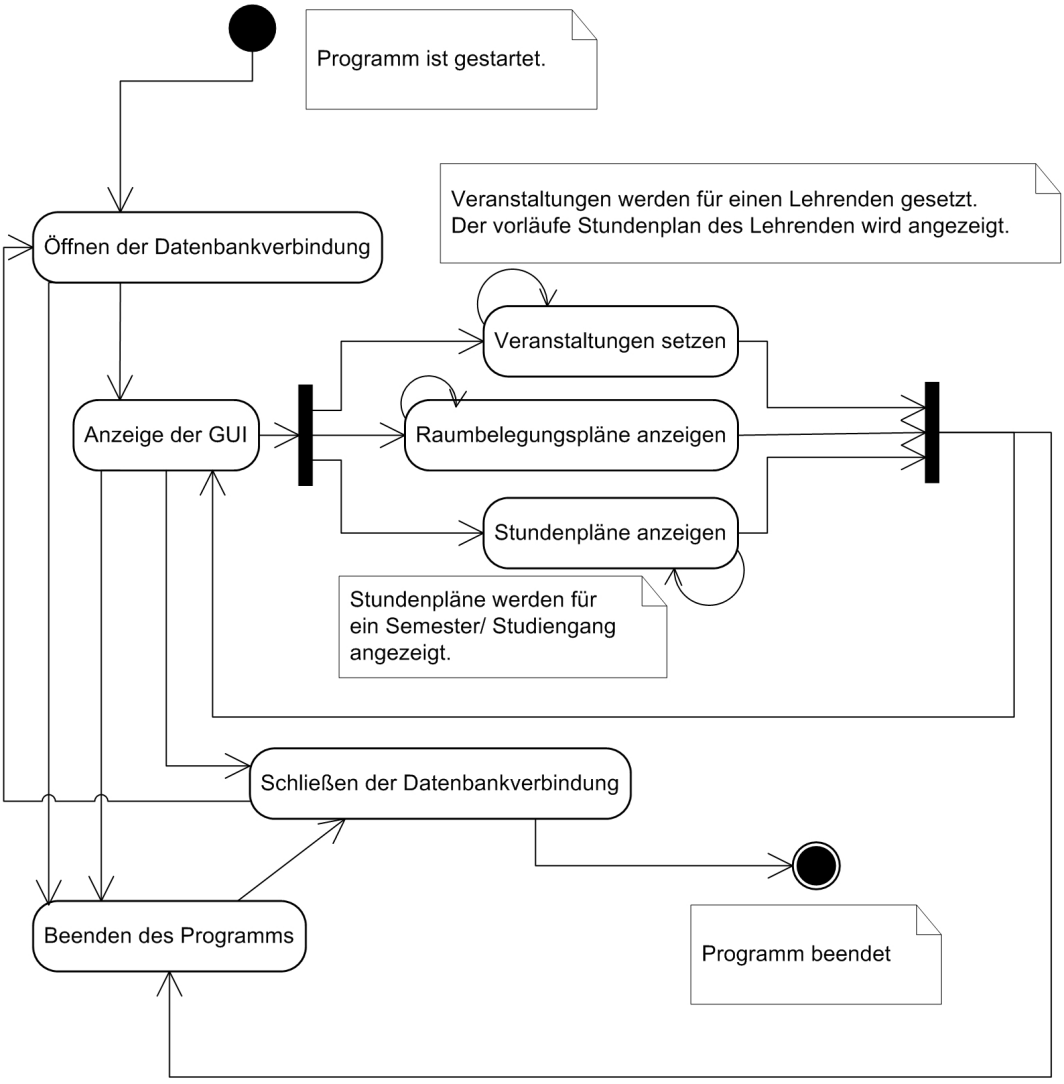

Abbildung 2: Allgemeiner Programmablauf [stateDiagram1.jpg]

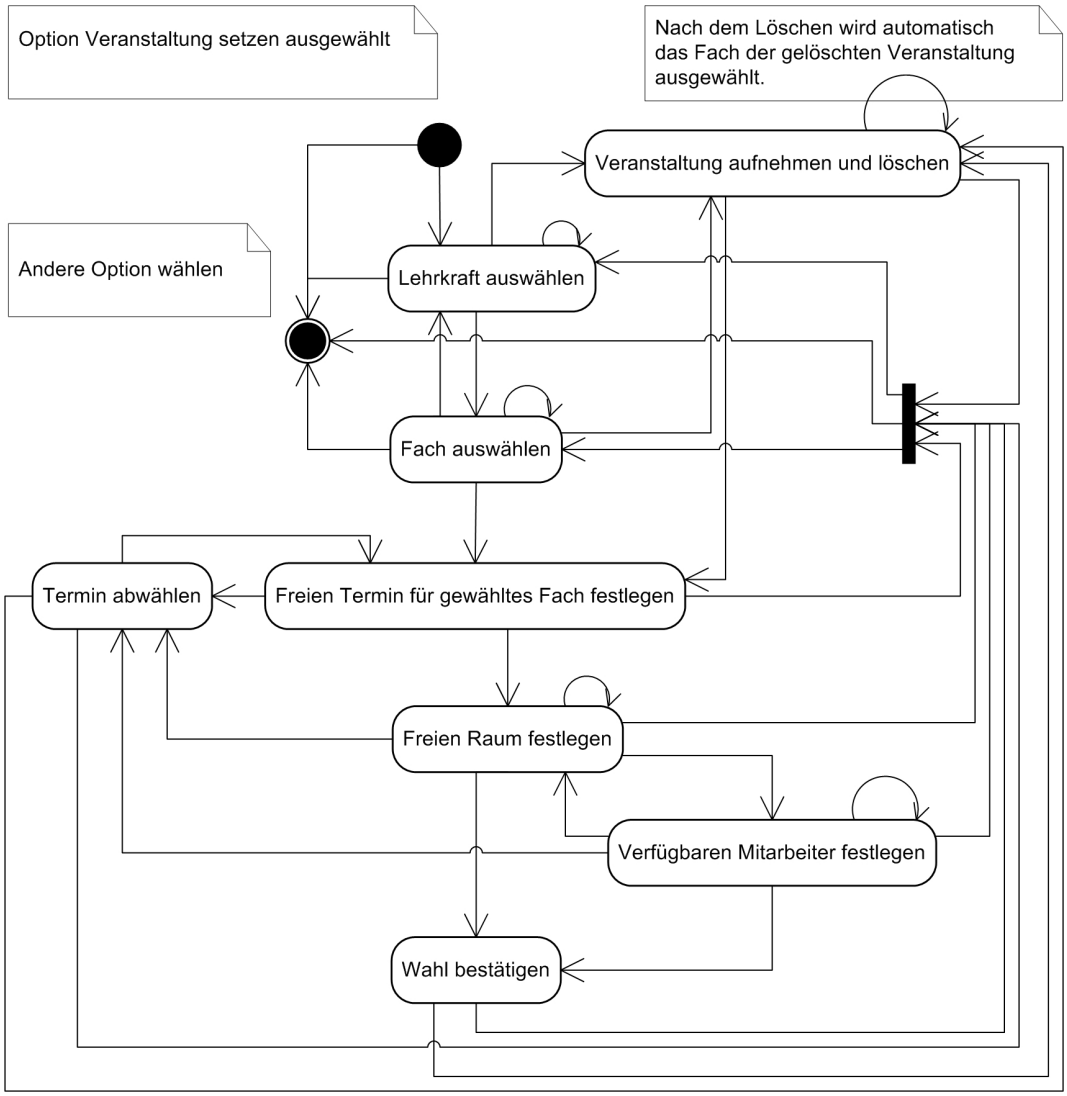

Abbildung 3: Festlegen einer Veranstaltung [stateDiagram2.jpg]

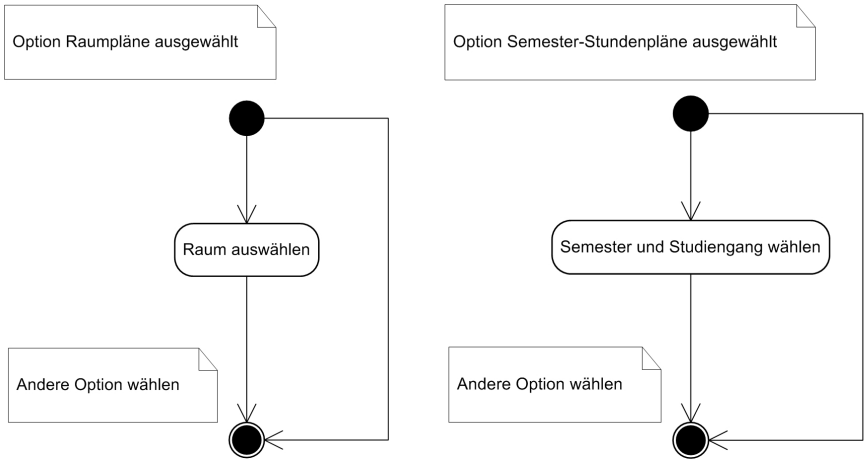

Abbildung 4: Raumplan (links), Semesterplan (rechts) anzeigen [stateDiagram3.jpg, stateDiagram4.jpg]

3.3 Klassendiagramme

Die Klassendiagramme werden in zwei Pakete unterteilt. Das Paket *db* beinhaltet die Klassen, die als Containerklassen für die Informationen aus der Datenbank verwendet werden. Das Paket *control* beinhaltet die Klassen, die zur Steuerung und Datenabfrage, bzw. Manipulation dienen. Die grafische Oberfläche wird von der Logik getrennt und durch das Interface `TimetableGUI` repräsentiert. Die Klassen, die für die grafische Oberfläche verwendet werden sind hier nicht als Klassendiagramm dargestellt.
Die für die Containerklassen zuständigen Broker werden hier nicht näher dargestellt. Sie dienen als Schnittstelle zwischen dem Programm und der Datenbank. In dieser Schnittstelle werden die Entities (DB) in konkrete Java-Objekte umgesetzt. Wie Broker verwendet werden, kann in Anbindung einer relationalen Datenbank an Java [1.5MB] nachgelesen werden. Die Containerklassen werden als innere Klassen der Broker modelliert, so daß eine Objekterzeugung nur über die Methoden der Brokerklassen möglich ist.

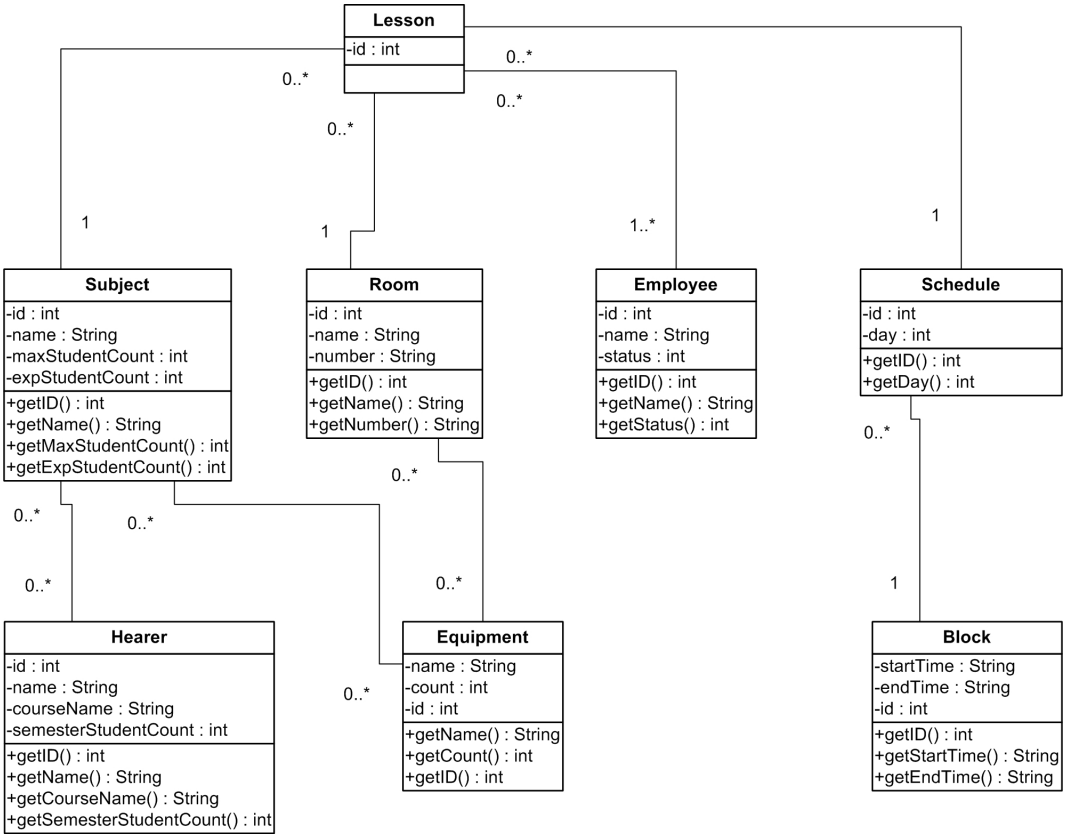

Abbildung 5: Paket *db* [ClassDiagramDB.jpg]

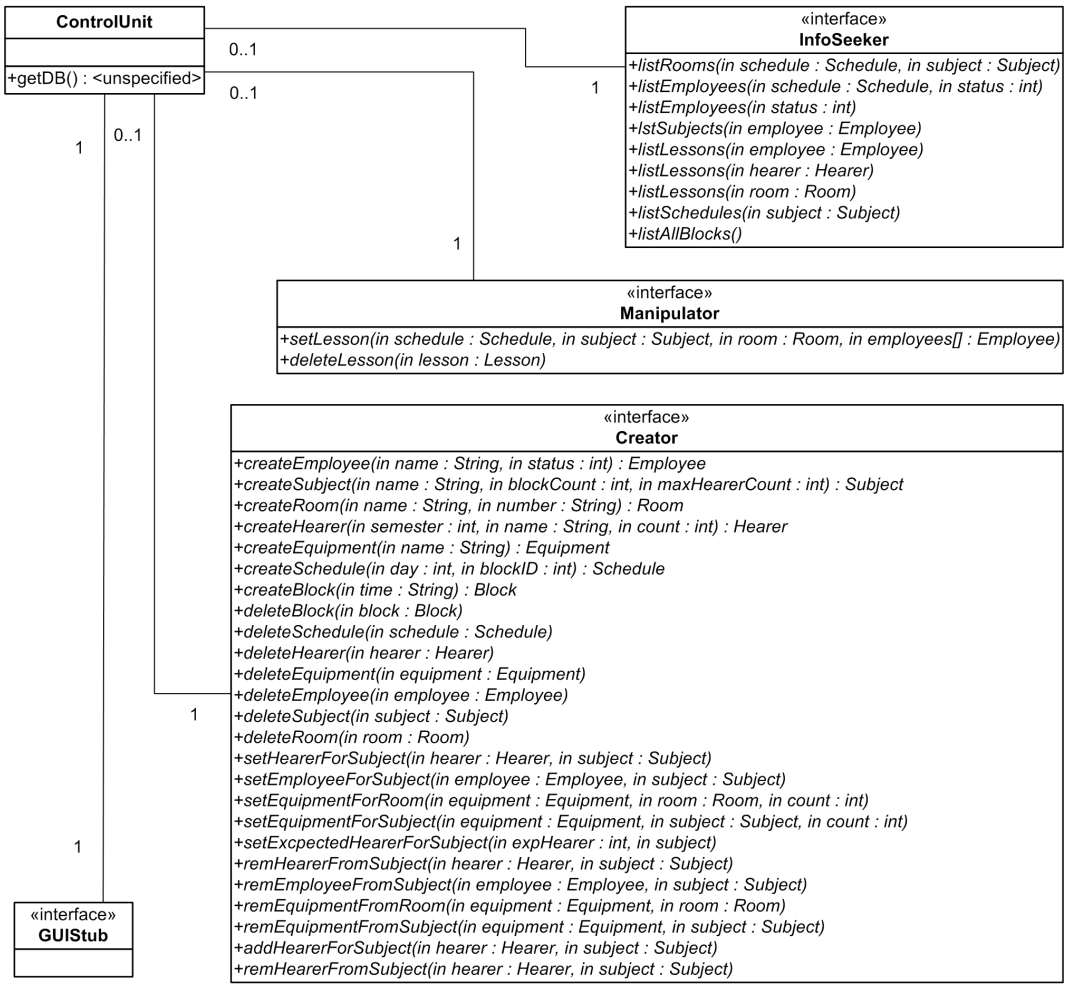

Abbildung 6: Paket *control* [ClassDiagramControl.jpg]

3.4 Interaktionsdiagramme

Das Interaktionsdiagramm beschränkt sich hier auf den Hauptanwendungsfall *Veranstaltung festlegen*. Die Interaktionsdiagramme der anderen Anwendungsfälle besitzen entweder eine ähnliche Struktur oder sind trivial.

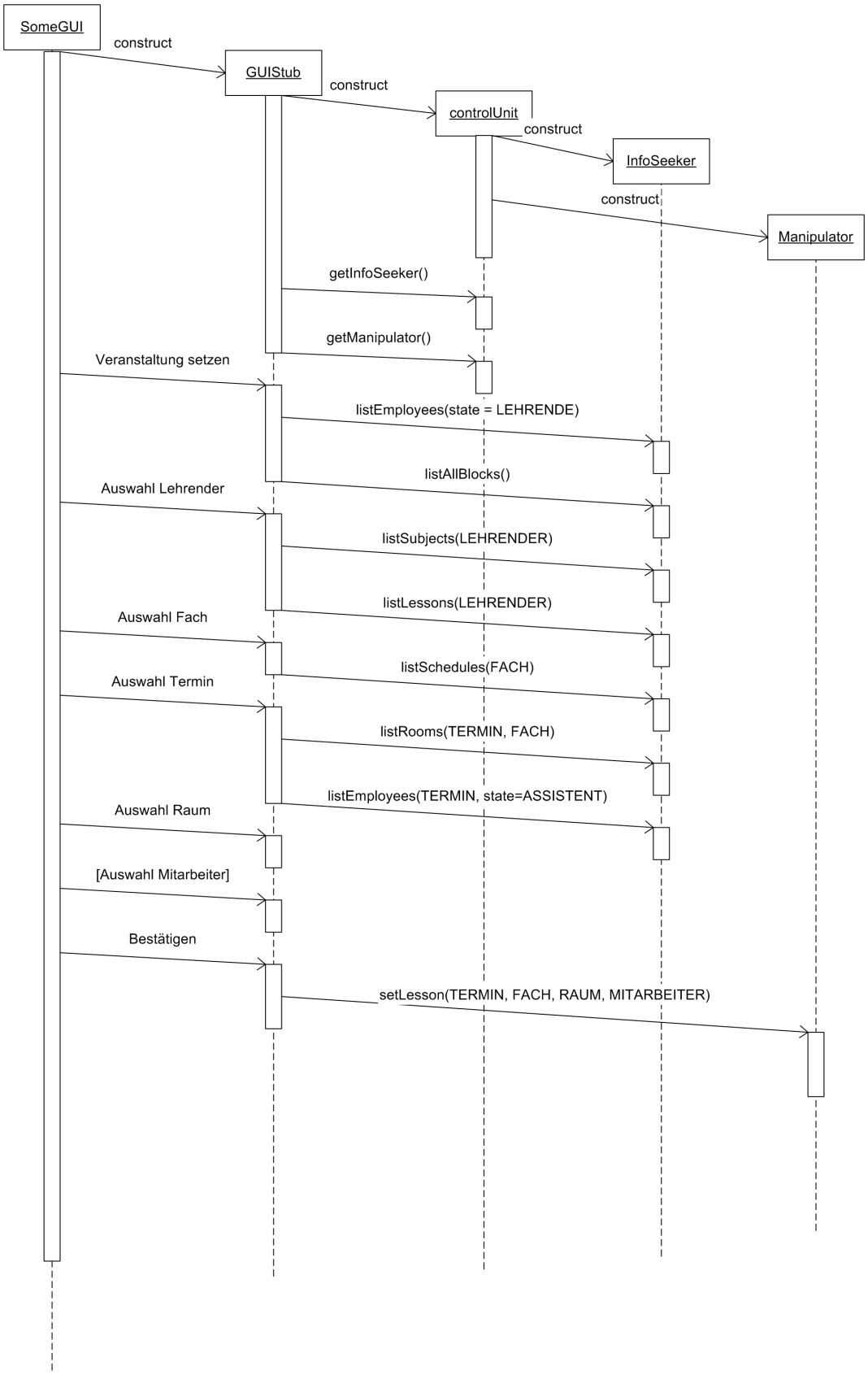

Abbildung 7: Festlegen einer Veranstaltung [SequenceDiagram.jpg]

3.5 Analyse der Datenbank

3.5.1 Entity Relationship Model

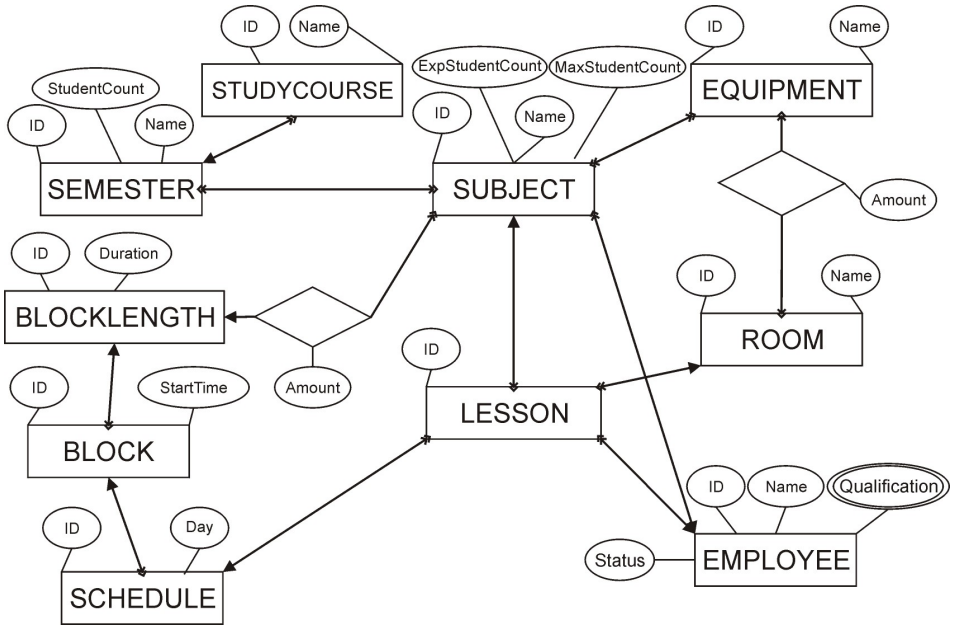

Abbildung 8: ursprüngliches ERM der Datenbank [ERM.jpg]

Dieses ER Modell gibt die Struktur der ersten Datenbank wieder. Im Laufe des Entwurfs / Anfang der Implementierung ergaben sich jedoch einige Punkte die durch die Datenbank nur schlecht oder gar nicht repräsentiert werden konnten. Wir fügten deshalb das Entity *Hearer* und das Beziehungsattribut *mandatory* in die Datenbank ein (siehe Abbildung 3.5.1).

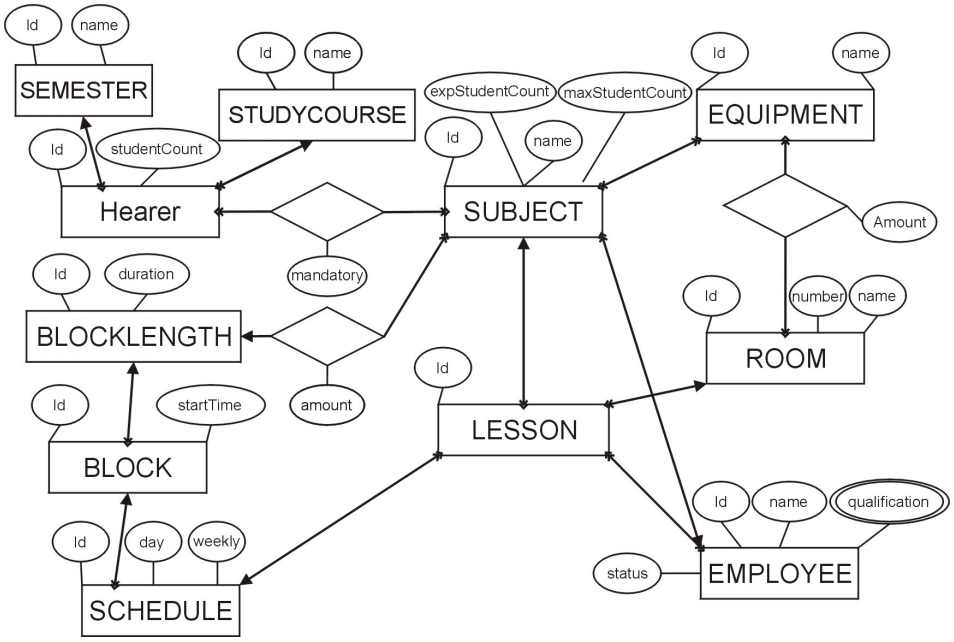

Abbildung 9: ERM der Datenbank [ERM.jpg]

3.5.2 Data Dictionary

```
Data Dictionary

Entity-Typ LESSON
        Attribute:              id
        Primärschlüssel:        id
Entity-Typ SCHEDULE
        Attribute:              id, day, weekly
        Primärschlüssel:        id
Entity-Typ BLOCK
        Attribute:              id, startTime
        Primärschlüssel:        id
Entity-Typ BLOCKLENGTH
        Attribute:              id, duration
        Primärschlüssel:        id
Entity-Typ STUDYCOURSE
        Attribute:              id, name
        Primärschlüssel:        id
Entity-Typ SEMESTER
        Attribute:              id, name
        Primärschlüssel:        id
Entity-Typ Hearer
        Attribute:              id, studentCount
        Primärschlüssel:        id
Entity-Typ EMPLOYEE
        Attribute:              id, name, status, qualification*=(id, name)
        Primärschlüssel:        id
Entity-Typ ROOM
        Attribute:              id, name, number
        Primärschlüssel:        id
Entity-Typ EQUIPMENT
        Attribute:              id, name
        Primärschlüssel:        id
Entity-Typ SUBJECT
        Attribute:              id, name, expStudentCount, maxStudentCount
        Primärschlüssel:        id
Beziehungstyp Hearer_Semester
        Beteiligte Entity-Typen: HEARER, SEMESTER
        Komplexitätsgrad:       cn:1
Beziehungstyp Hearer_Studycourse
        Beteiligte Entity-Typen: HEARER, STUDYCOURSE
        Komplexitätsgrad:       cn:1
Beziehungstyp  Hearer_Subject
        Beteiligte Entity-Typen: HEARER, SUBJECT
        Komplexitätsgrad:       cn:cn
Beziehungstyp Blocklength_Subject
        Beteiligte Entity-Typen: BLOCKLENGTH, SUBJECT
        Komplexitätsgrad:       1:cn
Beziehungstyp Lesson_Subject
        Beteiligte Entity-Typen: LESSON, SUBJECT
        Komplexitätsgrad:       cn:1
Beziehungstyp Equipment_Subject
        Beteiligte Entity-Typen: EQUIPMENT, SUBJECT
        Komplexitätsgrad:       cn:cn
Beziehungstyp Employee_Subject
        Beteiligte Entity-Typen: EMPLOYEE, SUBJECT
```

```
        Komplexitätsgrad:       1:cn
Beziehungstyp Equipment_Room
        Beteiligte Entity-Typen: EQUIPMENT, ROOM
        Komplexitätsgrad:       cn:cn
Beziehungstyp Lesson_Room
        Beteiligte Entity-Typen: LESSON, ROOM
        Komplexitätsgrad:       cn:1
Beziehungstyp Lesson_Employee
        Beteiligte Entity-Typen: LESSON, EMPLOYEE
        Komplexitätsgrad:       cn:n
Beziehungstyp Lesson_Schedule
        Beteiligte Entity-Typen: LESSON, SCHEDULE
        Komplexitätsgrad:       cn:1
Beziehungstyp Schedule_Block
        Beteiligte Entity-Typen: SCHEDLUE, BLOCK
        Komplexitätsgrad:       cn:1
Beziehungstyp Block_Blocklength
        Beteiligte Entity-Typen: BLOCK, BLOCKLENGTH
        Komplexitätsgrad:       cn:1
```

[DataDictionary.txt]

3.6 Prototype des GUI

Der hier gezeigte Prototyp wurde mit Borlands JBuilder 4 erstellt und zeigt die Funktionsweise der grafischen Oberfläche. Anhand des nachfolgenden Screenshots wird der Aufbau erklärt.

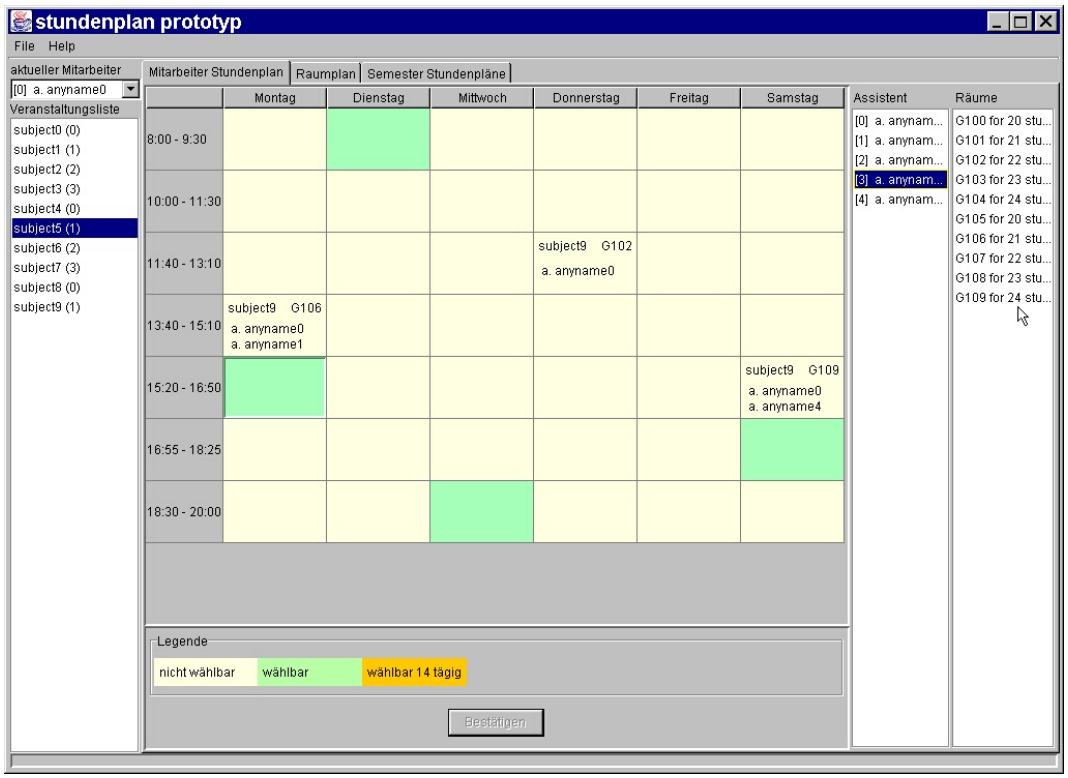

Abbildung 10: Prototype des GUI [prototyp.jpg]

In der Combobox links oben wird der Lehrende ausgewählt, der an der Reihe ist seine Veranstaltungen im Stundenplan zu setzen. Nach der Auswahl eines Lehrenden wird die darunterstehende Liste mit den noch zu vergebenden Fächern angezeigt. In der Liste steht der Fachname und in Klammern die Anzahl der Blöcke, die noch zu verteilen sind.
Wird in dieser Liste ein Fach gewählt, so erscheinen auf der Stundenplantabelle (Mitte) alle freien Termine in grün. Termine die nur 14tägig wählbar sind erscheinen in orange. Bewegt man den Mauszeiger nun über eines der grünen (orangenen) Terminfelder, so werden in der Mitarbeiterliste, bzw. Raumliste die verfügbaren Resourcen angezeigt. Entscheidet man sich für einen Termin, so wählt man diesen mit einem Mausklick aus. Dies führt dazu, daß die Raum und Mitarbeiterlisten fixiert werden. In diesen kann nun ein Mitarbeiter (optional) und ein Raum ausgewählt werden. Nach der Wahl des Raumes wird der Knopf **Bestätigen** freigeschaltet. Ist man mit der Wahl unzufrieden kann man durch erneutes Anklicken des Termins diesen wieder abwählen. Nach dem Bestätigen kann ein neuer Lehrender, bzw. ein neues Fach gewählt, oder aber weitere Blöcke des bereits gewählten Fachs gesetzt werden.

4 Entwurf

4.1 Systementwurf

Datenhaltung
Die Datenhaltung des Programms geschieht auf Grund der Komplexität und Effizienz der Abfragen in einer Datenbank. Dies sichert auch die Konsistenz der Daten und schränkt den Datenverlust bei Abstürzen stark ein.

Verteilung
Das Programm soll als Einzelplatzversion laufen, aber auch mit Datenbanken auf entfernten Rechnern arbeiten können.

Module
Das Programm wird in verschiedene Module eingeteilt, die spezielle Aufgaben erfüllen.

- **db**
  Umsetzung der Datenbankinformationen in Containerklassen.
- **control**
  Steuerung des Programmflusses und Bereitstellen von Informationen aus der Datenbank.
- **gui**
  Vom Programmfluss unabhängige Anzeige der Daten. Bereitstellung von interaktiven Elementen zur Steuerung des Programmflusses (siehe Kapitel 4.5 Umsetzung der Zustandsdiagramme als Automat)

4.2 Umsetzung der Assoziationen

Aus Platzgründen wurden in den Klassendiagrammen die Assoziationen nicht benannt. Deshalb geben wir hier zur jeder Klasse die Assoziationen und deren Umsetzung in folgenden Format an:

vonKlasse[Multiplizität] - nachKlasse[Multiplizität]

Die Beschreibung der Umsetzung geschieht hier richtungsabhängig.

4.2.1 Assoziationen im Paket db

Die Assoziationen der Broker-Klassen werden hier nicht aufgelistet, da alle Broker die gleiche Umsetzung haben. Die Beziehungen zu den Containerklassen werden über interne Klassen realisiert (siehe Kapitel 4.4). Die Beziehung zur ControlUnit wird über ein private Attribut realisiert.

Block enthält folgende Assoziationen:

- Block[1] - Schedule[0..*]
  wird nicht umgesetzt.

Employee enthält folgende Assoziationen:

- Employee[1..*] - Lesson[0..*]
  wird nicht umgesetzt.

Equipment enthält folgende Assoziationen:

- Equipment[0..*] - Room[0..*]
  wird nicht umgesetzt.
- Equipment[0..*] - Subject[0..*]
  wird nicht umgesetzt.

Hearer enthält folgende Assoziationen:

- Hearer[0..*] - Subject[0..*]
  wird nicht umgesetzt.

Lesson enthält folgende Assoziationen:

- Lesson[0..*] - Subject[1]
  Als private Attribut mit Accessormethode.
- Lesson[0..*] - Employee[1..*]
  Als private Array (TYP: Employee) mit Accessormethode.
- Lesson[0..*] - Room[1]
  Als private Attribut mit Accessormethode.
- Lesson[0..*] - Schedule[1]
  Als private Attribut mit Accessormethode.

Room enthält folgende Assoziationen:

- Room[0..*] - Equipment[0..*]
  Als private Array (TYP: Equipment) mit Accessormethode.
- Room[1] - Lesson[0..*]
  wird nicht umgesetzt.

Schedule enthält folgende Assoziationen:

- Schedule[0..*] - Block[1]
  Als private Attribut mit Accessormethode.
- Schedule[1] - Lesson[0..*]
  wird nicht umgesetzt.

Subject enthält folgende Assoziationen:

- Subject[0..*] - Equipment[0..*]
  Als private Array (TYP: Equipment) mit Accessormethode.
- Subject[0..*] - Hearer[0..*]
  Als private Array (TYP: Hearer) mit Accessormethode.
- Subject[1] - Lesson[0..*]
  wird nicht umgesetzt.

4.2.2 Assoziationen im Paket control

Auch hier werden die Broker-Klassen nicht aufgeführt, da die Beziehung für alle Broker über ein Singleton-Muster realisiert wird.

ControlUnit enthält folgende Assoziationen:

- ControlUnit[0..1] - InfoSeeker[1]
  Kann über ein Singleton-Muster im InfoSeeker realisiert werden.
- ControlUnit[0..1] - Manipulator[1]
  Kann über ein Singleton-Muster im Manipulator realisiert werden.
- ControlUnit[0..1] - Creator[1]
  Kann über ein Singleton-Muster im Creator realisiert werden.
- ControlUnit[1] - GUIStub[1]
  wird nicht umgesetzt.
- ControlUnit[1] - Database[1]
  wird als private Attribut mit Accessormethode umgesetzt.

InfoSeeker enthält folgende Assoziationen:

- InfoSeeker[1] - ControlUnit[0..1]
  als private Attribut.

Manipulator enthält folgende Assoziationen:

- Manipulator[1] - ControlUnit[0..1]
  als private Attribut.

Creator enthält folgende Assoziationen:

- Creator[1] - ControlUnit[0..1]
  als private Attribut.

GUIStub enthält folgende Assoziationen:

- GUIStub[1] - ControlUnit[1]
  als private Attribut.

4.2.3 Assoziationen im Paket gui

Enthält nur eine Assoziation zur GUIStub, die als private Attribut umgesetzt wird.

4.3 Singleton Muster bei Broker-Klassen

Da es nicht sinnvoll ist, mehrere verschiedene Exemplare einer Broker-Klasse auf die Datenbank zugreifen zu lassen, wird das Singleton-Muster verwendet. So wird sichergestellt, daß in einem laufenden Programm nur ein Exemplar des Brokers existiert. Dies erleichert auch den synchronisierten Zugriff. Synchronisation ist zwar nicht erforderlich, da wir nur einen laufenden Prozess haben, aber unter dem Gesichtspunkt der Erweiterbarkeit bringt es enorme Vorteile.

4.4 Containerklassen als innere Klassen der Broker

Die Containerklassen werden als private innere Klassen der jeweiligen Broker implementiert. Dies stellt sicher, daß alle erzeugten Objekte auch tatsächlich gültige Daten enthalten. Es wird nicht geprüft ob bereits erzeugte Objekte auf Grund von Datenmanipulationen ihre Gültigkeit verlieren.

4.5 Umsetzung der Zustandsdiagramme als Automat

In der Klasse GUIStub werden die Zustandsdiagramme der Analyse als Automat umgesetzt. Dies dient dazu eine Graphische Oberfläche dazu zu zwingen die einzelnen Schritte der Stundenplanerzeugung genau wie in der Analyse festgelegt abzuarbeiten. Dies vermeidet unsinnige Eingaben. Versucht die GUI einen nicht zugelassenen Zustand zu betreten oder eine im aktuellen Zustand nicht zugreifbare Methode aufzurufen, so wird eine Exception geworfen.

4.6 Normalisierung der Datenbank

Die Normalisierung des Relationenschemas dient dazu die Daten so in Tabellen anzuordenen, daß durch die Tabellenstruktur (Primärschlüssel, Fremdschlüssel) eine Konsistenzsicherung, durch die Datenbank selbst, erfolgen kann.

4.6.1 Unnormalisiertes Relationenschema

```
Semester = (semesterId, semesterName)
Studycourse = (courseId, courseName)
Hearer = (hearerId, studentCount, semesterId,courseId)
Blocklength = (blocklengthId, duration)
Block = (blockId, startTime, blocklengthId)
Schedule = (scheduleId, day, weekly, blockId)
mit day = (dayId, dayName, used)
Employee = (employeeId, employeeName, status, qualification*)
mit qualification = (qualificationId, qualificationName)
Room = (roomId, roomName, roomNumber)
Equipment = (equipmentId, equipmentName)
Subject = (subjectId, subjectName, expStudentCount, maxStudentCount, teacherId)
Lesson = (lessonId, subjectId, roomId, scheduleId)

Equipment_Room = (roomId, equipmentId, amount)
Blocklength_Subject = (subjectId, blocklengthId, amount)
Hearer_Subject = (hearerId, subjectId, mandatory)
Equipment_Subject = (subjectId, equipmentId)
Employee_Lesson = (employeeId, lessonId)
```

4.6.2 1 Normalform

Beseitigen von Wiederholungsgruppen und Zusammensetzungen.

```
Semester = s.o.
Studycourse = s.o.
Hearer = s.o.
Blocklength = s.o.
Block = s.o.
Schedule = (scheduleId, dayId, dayName, used, weekly, blockId)
Employee = (employeeId, employeeName, status )
Employee_Qualifikation = (employeeId, qualificationId, qualificationName)
Room = s.o.
Equipment = s.o.
Subject = s.o.
Lesson = s.o.

Equipment_Room = s.o.
Blocklength_Subject = s.o.
Hearer_Subject = s.o.
Equipment_Subject = s.o.
Employee_Lesson = s.o.
```

4.6.3 2 Normalform

Beseitigen der Abhängigkeit von nicht Primärschlüssel-Attributen und echten Teilen des Primärschlüssels.

Semester = s.o.
Studycourse = s.o.
Hearer = s.o.
Blocklength = s.o.
Block = s.o.
Schedule = (scheduleId, dayId, weekly, blockId)
Day = (dayId, dayName, used)
Employee = s.o.
Employee_Qualifikation = (employeeId, qualificationId)
Qualification = (qualificationId, qualificationName)
Room = s.o.
Equipment = s.o.
Subject = s.o.
Lesson = s.o.

Equipment_Room = s.o.
Blocklength_Subject = s.o.
Hearer_Subject = s.o.
Equipment_Subject = s.o.
Employee_Lesson = s.o.

4.6.4 3 Normalform

Beseitigen der Abhängigkeiten von Nicht-Primärschlüssel-Attributen.

Ist bereits erfüllt!

4.7 Das vollständige Data Dictionary

Semester

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
semesterId	Autowert, Primärschlüssel	Long Integer	No	Yes without duplicates
semesterName		String[50]	No/ No	No

Hearer

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
semesterId		Long Integer	No	Yes
courseId		Long Integer	No	Yes
studentCount		Long Integer	Yes	No
hearerId	Autowert, Primärschlüssel	Long Integer	No	Yes without duplicates

Hearer_Subject

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert

hearerId	Primärschlüssel	Long Integer	No	Yes
subjectId	Primärschlüssel	Long Integer	No	Yes
mandatory	Ob dieses Fach parallel zu anderen Fächern angeboten werden kann. Dies bezieht sich jeweils auf die Hearer. 0 = Wahlpflicht 2 = noch eine weiteres dieser Kategoprie 4 = nur dieses Fach	byte	No	No

Studycourse

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
courseId	Autowert, Primärschlüssel	Long Integer	No	Yes without duplicates
courseName		String[50]	No, Yes	No

Blocklength

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
blocklengthId	Autowert, Primärschlüssel	Long Integer	No	Yes without duplicates
duration		Integer	No	No

Block

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
blockId	Autowert, Primärschlüssel	Long Integer	No	Yes without duplicates
startTime		String[8]	No, No	No
blocklengthId		Long Integer	No	Yes

Schedule

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
scheduleId	Primärschlüssel	Long Integer	No	Yes without duplicates
dayId		Long Integer	No	Yes
blockId		Long Integer	No	Yes
weekly	1 = ungerade Woche, 2 = gerade Woche, 3 = wöchentlich	Long Integer	No	Yes

Day

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
dayId	Autowert, Primärschlüssel	Long Integer	No	Yes without duplicates
dayName		String[20]	No, Yes	No
used		Boolean	No	Yes
	Auswahl ob an diesem Tag Versantaltungen gesetzt werden können			
	Tage stehen konstant in der Datenbank.			

Room

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
roomId	Autowert, Primärschlüssel	Long Integer	No	Yes without duplicates
roomName		Strring[50]	No, Yes	No
roomNumber		String[15]	No, No	Yes without duplicates

Equipment_Room

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
roomId	Primärschlüssel	Long Integer	No	Yes
equipmentId	Primärschlüssel	Long Integer	No	Yes
amount	Anzahl des Equipments	Long Integer	No	Yes

Equipment

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
equipmentId	Autowert, Primärschlüssel	Long Integer	No	Yes without duplicates
equipmentName		String[50]	No,Yes	Yes without duplicates

Equipment_Subject

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
subjectId	Primärschlüssel	Long Integer	No	Yes
equipmentId	Primärschlüssel	Long Integer	No	Yes

Subject

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
subjectId	Autowert, Primärschlüssel	Long Integer	No	Yes without duplicates
subjectName		String[30]	No, No	No
expStudentCount		Long Integer	No	No
maxStudentCount		Long Integer	No	No
teacherId		Long Integer	No	No

Blocklegth_Subjects

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
blocklengthId		Long Integer	No	Yes
subjectId	Primärschlüssel	Long Integer	No	Yes without duplicates
amount	Wochenstunden in Blöcke	single,float	No	No

Employee

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
employeeId	Autowert, Primärschlüssel	Long Integer	No	Yes without duplicates
employeeName		String[12]	No, No	Yes without duplicates
status	0 = NN, 1 = Lehrender, 2 = Mitarbeiter, 3 = beides, 4 = Im Moment nicht verfügbar	byte	No	Yes

Employee_Lesson

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
employeeId	Primärschlüssel	Long Integer	No	Yes
lessonId	Primärschlüssel	Long Integer	No	Yes

Lesson

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
lessonId	Autowert, Primärschlüssel	Long Integer	No	Yes without duplicates
subjectId		Long Integer	No	Yes
roomId		Long Integer	No	Yes
scheduleId		Long Integer	No	Yes

Qualification

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
qualificationId	Autowert, Primärschlüssel	Long Integer	No	Yes without duplicates
qualificationName		String[50]	No, No	No

Employee_Qualification

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
qualificationId	Primärschlüssel	Long Integer	No	Yes
employeeId	Primärschlüssel	Long Integer	No	Yes

[DataDictionaryEntwurf.txt]

5 Implementation

Während der Implementationsphase kam es zu einigen Schwierigkeiten bei der Umsetzung der Schnittstellen für Infoseeker, Manipulator und Creator. Die im Entwurf dokumentierten interfaces beinhalten alle Schnittstellen, die mit Objekten (z.B. Employee, Block oder Schedule) bedient werden. Dies ließ sich durch die Umsetzung dieser Klassen als innere Klassen der Broker nur bedingt oder mit sehr hohem Implementierungsaufwand realisieren. Wir haben uns deshalb dafür entschieden die Schnittstelle zu erweitern, so daß nur die wirklich benötigten primitiven Datentypen an die Schnittstelle übergeben werden.

Das Klassenkonzept wurde ansonsten wie geplant umgesetzt. Einige Hilfsklassen, die zur Visualisierung der Daten dienen sind nicht in dem Entwurf erwähnt, da sich die Notwendigkeit erst bei der Implementierung der GUI herausstellte. Beispiele hierfür sind das ListModel und TableModel, die in der Klasse GUIStub.java als interne Klassen umgesetzt sind.

Zur weiteren Beschreibung der Implementation möchten wir auf die JAVADOC Dokumentation verweisen. Diese liegt als HTML und PDF Dokument vor. Hier ein Link auf die PDF Version:

Hier anklicken um die JavaDoc Dokumentation als PDF zu lesen

6 Tests

Die in der Machbarkeitsanalyse angesprochenen Tests wurden von uns, soweit Sie von uns durchführbar waren, auch durchgeführt. Die Tests, die unabhängige Testpersonen benötigen, können von uns nicht in der geplanten Zeit durchgeführt werden, da der Weihnachtsstreß alle potentiellen Tester in Beschlag nimmt. Diese Test können aber bei Bedarf nachgeliefert werden.

6.1 Testen der Bedienbarkeit

6.1.1 Anzahl der durchschnittlichen Clicks

Im maximalen Fall muß 11 mal geklickt werden. Diese 11 mal teilen sich auf in:

1. Lehrenden Liste scrollen
2. Lehrenden auswählen
3. Fach scrollen
4. Fach auswählen
5. Termin selektieren
6. Assistenten scrollen
7. Assistenten auswählen
8. Räume scrollen
9. Räume auswählen
10. Vierzehntägig auswählen
11. bestätigen.

Dieser Extremfall taucht allerdings fast nie bei der Bedienung auf.

Im minimalen Fall kommen wir mit 5 Clicks aus. Diese Teilen sich auf in:

1. Lehrenden auswählen
2. Fach auswählen
3. Termin selektieren
4. Räume auswählen
5. bestätigen.

Dieser Fall tritt weitaus häufiger auf als der maximale Fall.

Der geschätzte Durchschnittliche Fall besteht aus 6.25 Clicks pro Zuordnung. Wir kamen zu diesem Ergebnis durch folgende Überlegung:

1. Die Lehrendenliste
   muß bei jedem Durchlauf maximal einmal gescrollt werden, da nach dem scrollen wieder alle weiteren Lehrenden sichtbar sind. Da dies im Mittel nur 4.5 mal (bei 18 SWS und zwei Blöcken pro Durchlauf) der Fall ist können wir diesen Fall getrost mit einem Click bewerten.
2. Fach auswählen
   Die Fachliste ist für alle Lehrenden groß genug, da jeder Lehrende bei 18SWS nur max. 9 Fächer unterrichten kann. Also ist auch hier die Annahme mit einem Click gerechtfertigt.

3. Termin selektieren
   Da die Auswahl von Assistenten und Räumen bei Drag&Drop immer für den Termin angezeigt wird, über dem sich der Maus-Cursor befindet ist hier mit fast immer korrekter Wahl zu rechnen. Deshalb bewerten wir diesen Punkt auch nur mit einem Click.

4. Raum auswählen
   Wir gehen davon aus, daß im maximalen Fall doppelt so viele Räume zur Verfügung stehen, wie angezeigt werden können. Daraus folgt, das jedes zweite mal die Liste zur Auswahl gescrollt werden muß. Deshalb bewerten wir diese Aktion mit 1.5 Clicks.

5. Auswahl eines Assistenten
   Assistenten werden nur für Praktikas benötigt. Da diese im Verhältnis zu den Vorlesungen ohne Assistenten nur selten vorkommen bewerten wir sie hier mit 0.5 Clicks.

6. Auswahl von vierzehntägigen Veranstaltungen
   Gehen wir davon aus, daß nur jede vierte Veranstaltung 14 tägig stattfindet, so müssen wir für diese Auswahl im Mittel 0.25 Clicks rechnen. Die Auswahl wöchentlich ist ja der Standardwert.

7. bestätigen
   Mit einem Click, wie auch sonst?

Damit liegen wir in einem guten Mittelfeld zwischen Soll und Muß Wert.

6.1.2 Schachtelungstiefe der Menüs

Dies ist ein sehr erfreulicher Punkt, da wir vollkommen ohne Untermenüs auskommen beträgt die Schachtelungstiefe genau dem Wunschwert von 1.

6.1.3 Note der Anwender

Zählt zu den noch nicht durchgeführten Tests durch unabhängige Benutzer.

6.2 Erweiterbarkeit

Die Anzahl der im vorhandenen Code geänderten Zeilen. Dies war in den meisten Fällen gar nicht notwendig, die Erweiterungen ließen sich ohne Änderungen am vorhandenen Quellcode einbauen. In ein zwei Fällen war es notwendig eine Überprüfung an anderer Stelle vorzunehmen, was sich nur durch das Umkopieren von max 15 Zeilen Code bewerkstelligen ließ.

6.3 Zuverlässigkeit

Wurde mangels Testpersonen nicht durchgeführt.

6.4 Portierbarkeit

Noch nicht notwendig und nur für Test nicht angebracht.

6.5 Konsistenz

Nachdem die Entwicklung der SQL Befehle abgeschlossen war kam es zu keinen uns bekannten Inkonsistenzen. Der GUI wurden allerdings bei Fehlbenutzung bis zur Endphase hin noch Eingabe-Fehler mitgeteilt, die zu Inkonsistenzen führen würden.

6.6 Robustheit

Nach Abschluß der GUI Entwicklung kam es zu keinen von bemerkten Eingabefehlern. Die GUI stellt nur für den Benutzer wählbare Optionen bereit. Die Entwicklung eines Test-Roboters konnte nicht im Rahmen dieses Projektes durchgeführt werden.

6.7 Reaktions-Zeit

Nach anfänglichen Schwierigkeiten mit der Ausführungsgeschwindigkeit der SQL-Anweisungen wurden von uns die SQL Ausdrücke optimiert und zum schnelleren Zugriff Views in der Datenbank erzeugt. Die Broker puffern die Daten, soweit sie nur zum lesen genutzt werden. Dies brachte enorme Geschwindigkeitsvorteile. Die Ausführungsgeschwindigkeit ist von 3min auf geschätzte 600 ms bis zur Anzeige gesunken. Diese Zeiten werden auch bei einer fast vollständig gefüllten Datenbank noch erreicht. Damit liegen wir auch bei diesem Punkt über dem Soll-Wert.

6.8 Erlernbarkeit

Wurde mangels Testpersonen nicht durchgeführt.

6.9 Wartbarkeit

Fehler die Während der Programmentwicklung auftraten zählen nicht zum Test. Die Fehlerbehebung in der Testphase beschränkte sich meist auf ein paar Minuten bis maximal 2 Stunden.