

Studienplaner

Schriftliche Ausarbeitung



Gruppenarbeit für Programmieren II (SS 2014)

Leibniz Universität Hannover

Von

Wladimir Hannefeld

Niklas Kielblock

Michael Thies

Tutoren

Gerion Entrup

Sergej Wildemann

1. Aufgabenstellung

Aufgabe war es, ein Programm zur Planung des Informatik-Studiums anhand der aktuellen Prüfungsordnung zu schreiben. Dabei sollen die möglichen Veranstaltungen mit ihren Abhängigkeiten untereinander in einem Graph dargestellt werden, der vom Benutzer editiert werden kann, um die Veranstaltungen und ihre Beziehungen zu verändern. Aus diesen Daten soll automatisch, anhand weiterer Angaben, ein Studienplan generiert werden.

Sowohl die Veranstaltungen, als auch der Studienplan sollen per XML gespeichert und geladen werden können und in Form von Graph bzw. Tabelle als PDF-Datei exportiert werden.

2. Umsetzung

2.1 Aufteilung, Klassenstruktur

Wir haben uns schon recht früh entschieden, den Code grob nach dem Model-View-Controller-Pattern aufzuteilen. Das Model umfasst die gesamte Datenbasis und stellt Methoden zum Abfragen und Ändern von Daten bereit. Bei Änderungen führen die Klassen einige Validierungen durch und benachrichtigen verbundene Views und Controller, damit diese Änderungen in alle relevanten Teile der grafischen Oberfläche übertragen werden. Die Views sind für das Rendern der Daten in Swing und Weitergabe von Interaktionen an Controller verantwortlich, die Controller koordinieren den Programmfluss und verbinden die Komponenten. Algorithmen zur Datenverarbeitung und Im-/Export sind in einem `util`-Package untergebracht, arbeiten anhand von Modelobjekten und werden ebenso von Controllern verwendet.

Bedienungskonzept, Funktionsumfang und Datenmodel entwickelten wir gemeinsam, letzteres setzten wir auch als erstes um und erstellten eine Projektstruktur, um eine Basis für weitere Arbeiten zu haben. Danach teilten wir uns in folgende Bereiche auf:

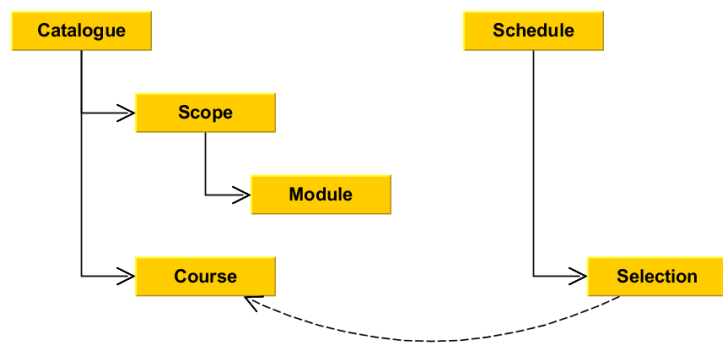
- Michael war für die Entwicklung der Algorithmen zum Im-/Export nach XML (und den dazugehörigen XML-Dokumentstrukturen), PDF-Export und Generierung und Validierung von Stundenplänen zuständig,
- Wladimir für das Design und die Umsetzung der grafischen Oberfläche mit all ihren Fenstern, Menüs, Dialogen und Tabellen, und
- Niklas für die JGraphX-basierte Graphkomponente, die Catalogues in einen Graph einträgt und mit minimalem Codeaufwand in obige eingebaut wurde.

So konnten wir diese Komponenten größtenteils einzeln und alleine entwickeln, ohne uns über deren innere Funktionsweise austauschen zu müssen. Gegen Ende der Entwicklungszeit halfen wir uns aber in größerem Umfang bei der Interaktion der Komponenten sowie der Fehlerbehebung und Implementierung letzter Features - zu dieser Zeit testeten wir das zusammengesetzte Programm und schauten daher auch vermehrt auf den internen Code aller Komponenten; uns hier noch streng daran zu halten, nur an den eigenen Komponenten zu entwickeln, hätte unsere Produktivität eingeschränkt.

Die Dokumentation schrieben wir gemeinsam mit einem kollaborativen Editor, wobei die Teile über einzelne Komponenten logischerweise größtenteils von denen geschrieben wurden, die für diese verantwortlich waren.

2.2 Klassenstruktur des Modells

Das Modell, das der Anwendung zu Grunde liegt, besteht aus insgesamt 6 Klassen und einigen Helfern. Das nebenstehende (stark vereinfachte) Diagramm veranschaulicht die wichtigsten Beziehungen zwischen ihnen.



Die Klasse `Catalogue` repräsentiert einen Modulkatalog mit

Kompetenzbereichen (`Scope`), Modulen (`Module`) und Veranstaltungen (`Course`). Dabei werden die `Courses` in einem Set des `Catalogues` gespeichert und enthalten selbst eine Referenz auf den `Scope` und das `Module` zu dem sie gehören. Parallel zu dieser Struktur gibt es ein zweites Modell aus einem Studienplan (`Schedule`), der auf einem bestimmten `Catalogue` basiert und beliebig viele `Selections` enthält. Jede solche `Selection` bedeutet die Planung eines bestimmten `Courses` in einem bestimmten Semester. Zusätzlich enthält der `Schedule` eine Liste von `Courses`, die der Nutzer zum Einplanen vorgemerkt hat.

Wie aus diesem Modell deutlich wird, können der `Catalogue` (mit allen Inhalten) und der `Schedule` (mit seinen `Selections`) unabhängig voneinander exportiert und import werden, allerdings ist ein `Schedule` immer von einem bestimmten `Catalogue` abhängig, der immer zuerst geladen werden muss.

2.3 Algorithmen

2.3.1 `AutoSchedule()` – Automatische Generierung des Studienplans

Der `AutoSchedule`-Algorithmus erhält einen (leeren oder nur teilweise gefüllten) Studienplan und einige weitere Parameter, die der Anwender angeben muss, und füllt unter Berücksichtigung der Vorgaben des zum Studienplan gehörigen Modul-Katalogs und den vom Anwender angegebenen Wünschen den Studienplan. Dabei werden alle Veranstaltungen eingeplant, die im Modul-Katalog als verpflichtend (`required`) markiert sind oder vom Anwender gewünscht wurden (`Schedule.desiredCourses`). Zudem werden alle Abhängigkeiten dieser Veranstaltungen ebenfalls eingeplant. Vom Benutzer bereits manuell eingetragene Veranstaltungen werden dabei beibehalten und bei der Planung berücksichtigt.

Der Anwender muss zum Starten des Algorithmus angeben, wie viele Leistungspunkte pro Semester maximal eingeplant werden sollen, in welchem Semester der Algorithmus mit dem automatischen Planen beginnen soll, wie wichtig empfohlene (nicht verpflichtende) Abhängigkeiten sind (und entsprechend beachtet werden sollen) und ob bereits belegte bzw. durchgefallene Kurse erneut eingeplant werden sollen.

Die grundsätzliche Vorgehensweise des Algorithmus:

- Alle bereits geplanten oder bestandenen oder (sofern gewünscht) belegten/durchgefallenen Veranstaltungen einer Map hinzufügen, die das früheste Semester angibt, in dem die Veranstaltung absolviert/geplant wurde
- Alle verpflichtenden und gewünschten Veranstaltungen (und rekursiv ihre Abhängigkeiten) einer Map hinzufügen, zusammen mit einem Score, der die Priorität beim Einplanen angibt

(Abhängig von der Anzahl und Art der Ebenen von Abhängigkeiten und der Leistungspunkte dieser Veranstaltung)

- Die Veranstaltungen (in einer Liste) absteigend nach dieser Priorität sortieren
- Im ersten zu planenden Semester beginnen und für jedes Semester:
 - Sofern die Liste nicht leer ist, am Anfang der Liste beginnen und für jede Veranstaltung
 - Überprüfen, ob im Semester noch Platz für sie ist, ihre Abhängigkeiten erfüllt sind und sie in diesem Semester angeboten wird
 - Wenn ja: Veranstaltung Einplanen und aus der Liste löschen,
 - Wenn nein: zur nächsten Veranstaltung gehen
 - Wenn das Ende der Liste erreicht: Zum nächsten Semester springen

Eine gesonderte Behandlung erfahren dabei empfohlene Abhängigkeiten: Sie werden zum einen bei der Priorisierung der Veranstaltungen weniger stark gewichtet und zum anderen beim Einplanen unter bestimmten Umständen (abhängig von der Angabe des Nutzers und dem aktuellen Semester) nicht beachtet.

2.3.2 IO - Ex- und Import nach/von XML

Der XML Export ist über einen `XMLStreamWriter` realisiert. Innerhalb der `IO.saveCatalogue()` und `IO.saveSchedule()` Methoden werden manuell alle Eigenschaften durchlaufen und als XML-Elemente in eine Datei geschrieben. Diese Methode bietet den Vorteil, dass sie eine genaue Kontrolle über das Export-Format zulässt und auch bei großen Datenmengen arbeitsspeicherschonend ist.

Zum Import wird ein SAX-Parser verwendet, der bei jedem XML-Event (öffnender oder schließender Tag, einzelne Zeichen, ...) eine Callback-Funktion aufruft. Auf diese Weise kann während des Einlesens der XML-Datei über die Klassen `SAXCatalogueHandler` bzw. `SAXScheduleHandler` die jeweilige Datenstruktur im Programm erzeugt werden.

2.3.3 PDF-Export

Zum PDF-Export verwenden wir die proprietäre Library `iText`. Diese bietet zahlreiche Möglichkeiten, um PDF-Dateien zu erstellen und zu modifizieren. Um den Graphen in eine PDF-Datei zu bannen, wird mit `iText` ein `Graphics2D`-Objekt erstellt, das `JGraphX` nutzen kann, um den Graphen zu rendern. Er wird somit direkt ohne Verluste mit den Methoden von `iText` in die PDF-Datei gerendert.

Zum Erstellen von Tabellen in der PDF-Datei bietet `iText` selbst einige Möglichkeiten, die wir nutzen, um den Studienplan zu exportieren.

2.4 Graph

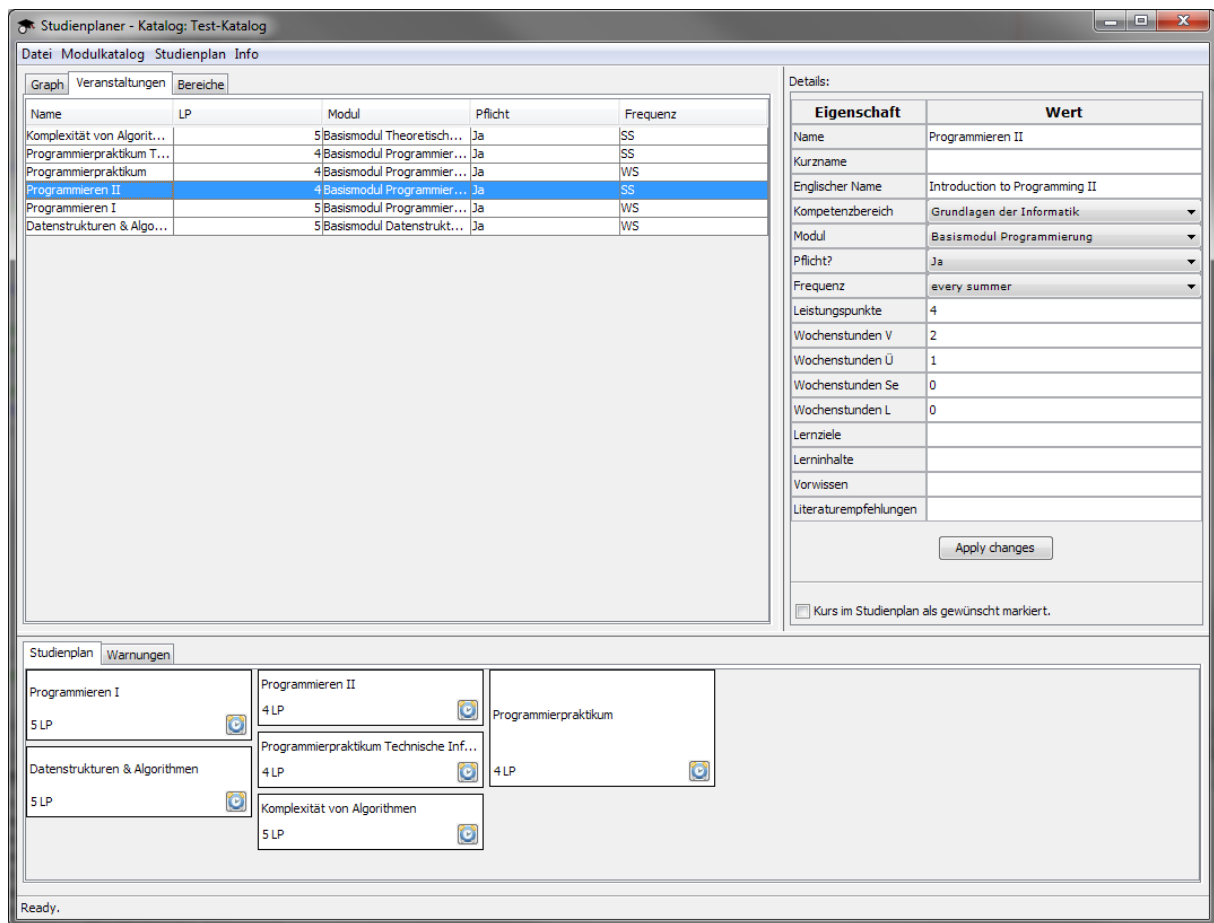
Den Graph haben wir, wie uns von unseren Tutoren nahegelegt wurde, mit der freien Library `JGraphX` implementiert. Dazu haben wir eine Swing-Komponente `GraphPanel` gebaut, die selbstständig den Graph einbindet und verwaltet.

Sie stellt eine Methode zum Laden eines Catalogue zur Verfügung, die für jeden Course und jede Dependency einen Vertex bzw. eine Edge im Graph erstellt und Zuordnungen zwischen den Graph- und Modelobjekten speichert. Außerdem observiert sie Änderungen am Model, die für die graphische Darstellung relevant sind, also das Hinzufügen oder als entfernt Markieren von Courses, das Hinzufügen und Entfernen von Dependencies sowie das Ändern darstellungsrelevanter Daten von Courses. Anders herum führt sie Handlungen, die im Graph passieren, auf das Model zurück und gibt

sie an einen Controller weiter, der ein zugehöriges Interface `IGraphController` implementiert (und die entsprechenden Änderungen an anderen Teilen der Oberfläche und im Model durchführen soll). Die erfassten Handlungen sind das Auswählen von Courses bzw. Dependencies durch Klick (und das Deselektieren durch Klick auf eine freie Stelle), das Umbenennen von Courses durch Doppelklick auf die Beschriftung, und das Erstellen von Dependencies. Letzteres geschieht durch Klicken und Ziehen eines Teils des abhängigen Course auf den Course, von dem dieser abhängig sein soll, und Einstellen der Parameter der Dependency in einem Popupidialog.

Für die Darstellung und Bedienung des Graph stellt sie in ihrem Konstruktor einige Einstellungen am internen `mxGraph`-Objekt (und dessen Swing-Komponente) ein, die hauptsächlich dazu dienen, es zu einem spezialisierten Graph zu machen (anstelle der standardmäßig in der Bibliothek aktivierten Funktionalität für einen allgemeinen Grapheditor). Courses werden als Vertices dargestellt, in dem Name, Leistungspunkte und Häufigkeit zu einer Aufschrift formatiert werden. Dependencies werden als Edges dargestellt, wobei als `required` markierte Dependencies als rote und als `parallel` markierte Dependencies als gestrichelte Pfeile vom abhängigen Course auf den Course, von dem voriger abhängt, angezeigt werden. Jedes Mal, wenn die Darstellung verändert wird (also auch beim initialen Laden eines Catalogues), wird der Graph anhand einer `JGraphX`-Methode hierarchisch geordnet.

3. Manual/Typischer Programmablauf



3.1 Programmstart

Bei Programmstart wird dem Anwender die Möglichkeit gegeben, einen bereits existierenden Modulkatalog zu laden (1) oder einen komplett neuen Modulkatalog zu erstellen (2).

- (1) Wird die Option gewählt, einen Katalog zu laden, wird ein "Öffnen"-Dialog für die Dateiauswahl angezeigt, sodass nach dem Bestätigen die ausgesuchte Datei geladen wird und deren Daten in die innere Objektstruktur übernommen werden.
- (2) Wird die Option gewählt, einen neuen Katalog zu erstellen, wird der Anwender nach einem Namen für diesen gefragt. Beim Bestätigen des Namens werden ein neuer Katalog, sowie ein leerer Studienplan erstellt.

3.2 Genereller Programmablauf

Nach dem Programmstart und Laden/Erstellen eines Katalogs steht dem Anwender eine Vielzahl an Möglichkeiten zur Verfügung, die eine Bearbeitung der Daten erlauben. Mögliche Operationen sind, den Komponenten zugeordnet, nachfolgend aufgelistet.

3.2.1 Die Menüleiste

Die Menüleiste bietet viele Optionen, die Im-/Export von Modul-Katalog und Studienplan, und deren Bearbeitung ermöglichen. Diese Möglichkeiten sind in vier Kategorien aufgeteilt, die in den Punkten **3.2.1.1** bis **3.2.1.3** näher erläutert werden.

3.2.1.1 Das "Datei"-Menü

Das "Datei"-Menü stellt Einträge für das Öffnen, Speichern und Exportieren der bearbeiteten Daten des Katalogs, Studienplans bzw. Graphen zur Verfügung. Bei der Wahl einer dieser Optionen erscheint ein Dialogfenster, in dem Name und Lade-/Speicherort der entsprechenden Dateien ausgewählt werden kann. Verschiedene Meldungen weisen gegebenenfalls auf auftretende Fehler hin oder fordern den Nutzer vor einem möglichen Datenverlust zum Speichern auf.

3.2.1.2 Das "Modulkatalog"-Menü

Das "Modulkatalog"-Menü stellt Funktionen zum Bearbeiten der Daten des Modul-Katalogs bereit, die in den Tabs "Veranstaltungen" und "Bereiche" angezeigt werden. Diese dienen zum Erstellen bzw. Löschen von Veranstaltungen, Kompetenzbereichen und Modulen. Für das Löschen eines dieser Objekte muss dieses zunächst ausgewählt werden. Nach einem weiteren Bestätigen des Löschvorgangs kann die betreffende Komponente entfernt werden. Beim Hinzufügen von Objekten muss nur darauf geachtet werden, dass ein Kompetenzbereich markiert sein muss, sofern das Hinzufügen eines Moduls gewünscht ist.

Darüber hinaus bietet dieses Menü die Möglichkeit, in der Graph-Ansicht erstellte Abhängigkeiten zu löschen, nachdem sie selektiert wurden.

3.2.1.3 Das "Studienplan"-Menü

Das "Studienplan"-Menü beinhaltet Optionen zur Modifikation des Studienplans:

- (1) Die automatische Studienplan-Erstellung wird anhand der Veranstaltungstabelle durchgeführt. Dabei werden die Veranstaltungen, die gemäß Modul-Katalog verpflichtend sind, und diejenigen, die als gewünscht markiert wurden, in die betreffenden Semester des Studienplans einsortiert. (vgl. **2.3.1**)
Dabei gibt es im dazugehörigen Dialogfenster weitere Möglichkeiten, um den Algorithmus zu konfigurieren: Etwa, ob bereits belegte Veranstaltungen ohne abgeschlossene Prüfung erneut eingeplant werden sollen oder welche Priorität empfohlene (nicht verpflichtende) Abhängigkeiten zwischen Kursen haben.
- (2) Die Validierung des Studienplans ermöglicht es, Fehler in der Planung (Nicht beachtete Abhängigkeiten, nicht angebotene Veranstaltungen, nicht erreichte LP-Grenzen) zu finden und diese sowohl in der Studienplan-Ansicht selbst, als auch in dem "Log"-Tab anzuzeigen bzw. aufzulisten.
- (3) Die weiteren Menüeinträge können zum manuellen Hinzufügen, Verschieben oder Entfernen der Veranstaltungen in der Studienplan-Ansicht benutzt werden. Für das Verschieben und Entfernen muss die entsprechende Veranstaltung im Studienplan markiert werden, für das Hinzufügen die Veranstaltung im "Veranstaltungen"-Tab oder im Graphen.

3.2.2 Die Graph-Ansicht

Die Graph-Ansicht ermöglicht es, Veranstaltungen (durch Doppelklicken) umzubenennen und die Abhängigkeiten zwischen den Kursen festzulegen, indem man auf die Mitte eines Kurses klickt und den erscheinenden Pfeil auf den Kurs zieht, von dem der aktuelle abhängt. Daraufhin öffnet sich ein Fenster, in dem man die Eigenschaften der gerade erstellten Abhängigkeit festlegen kann. Durch das auswählen einer Veranstaltung werden ihre Details im Bereich auf der rechten Seite angezeigt und können dort ebenfalls editiert werden.

3.2.3 Die Studienplan-Ansicht

Der aktuelle Studienplan wird im unteren Bereich des Fensters angezeigt, wo es möglich ist, einen Studienplan zusammenzustellen, indem man manuell Veranstaltungen aus der Liste hinzufügt, entfernt oder verschiebt, oder den automatischen Algorithmus verwendet. (vgl. **3.2.1.3**) Zudem ist es möglich über ein Icon mit Dropdown-Menü den Status der Planung zu ändern. Dabei wird unterschieden zwischen Veranstaltungen, die aktuell geplant sind und solchen die bereits besucht wurden (ohne abgeschlossene Prüfung, mit bestandener oder nicht bestandener Prüfung). Diese Angaben werden auch vom Planungsalgorithmus akzeptiert.

Auf diese Weise kann man beispielsweise seinen bisherigen Studienverlauf eintragen und mit dem automatischen Planungsalgorithmus die weiteren Semester planen.

3.2.4 Die Details-Ansicht

Die Details-Ansicht zeigt beim Auswählen eines Katalog-Objekts (Veranstaltung, Abhängigkeit, Kompetenzbereich, Modul) in einer Liste oder im Graphen dessen Eigenschaften (Name, ...) an und bietet die Möglichkeit zur Bearbeitung dieser Eigenschaften. Dazu müssen die gewünschten Felder ausgefüllt und der "Bestätigen"-Knopf am Ende der Liste ausgewählt werden.

3.3 Programmende

Nach der gewünschten Bearbeitung können der Katalog und der Studienplan abgespeichert werden. Darüber hinaus gibt es die Möglichkeit, den Studienplan und den Graphen jeweils im PDF-Format zu exportieren. (siehe Punkt **3.2.1.1**)

4. Known Issues und Erweiterungsmöglichkeiten

4.1 Graph

Die Anordnungsmethode, die wir verwenden, hat leider die Eigenschaft, den Graph sehr länglich darzustellen und dadurch eine große horizontale Scrollbar und vertikal viel verschwendeten Platz zu erzeugen. Dies sehen wir allerdings als Problem der Library an, da diese Methode ("horizontal hierarchical layout") dies per Design tun soll und die anderen Anordnungsmethoden, die JGraphX bereitstellt, sich entweder nicht für unseren Graphtypen eignen, oder schlichtweg nicht funktionieren. Letztere schlagen entweder sogar im (an sich voll funktionsfähigen) Editor-Codebeispiel, das mit der Library mitgeliefert wird, fehl, oder wurden dort gar nicht erst implementiert.

4.2 Weitere Fehler

- Studienplan wird nicht immer sofort gerendert
- Graph ohne Zoom schwer handhabbar

4.3 Mögliche Verbesserungen:

- Verbesserter Workflow durch Kontextmenüs, Drag-and-Drop und Tastatur-Shortcuts
- Einige Verbesserungen am Oberflächen-Layout und Dialogen

5. Kompilieren des Quellcodes

Kompilieren: `ant`

Jar erstellen: `ant jar`

Ausführen: `ant run`

Javadoc erstellen: `ant javadoc`

Ausführbarer Jar-File liegt nach der Erstellung (s.o.) im Verzeichnis `dist/`. Achtung: Benötigt die im Ordner `lib` beigelegten Jar-Bibliotheken.