# SPASS-TOOL

## Architekturdokumentation nach ARC-42

2022-06-16

Beate Arnold, Arthur Fieguth, Marie Bohnert

## Versionshistorie

| Datum | Bearbeiternamen | Titel |
|---|---|---|
| 04.06.2022 | Fieguth, Arnold, Bohnert | Einführung und Ziele, Aufgabenstellung |
| | Arnold | Qualitätsziele |
| | | |
| 07.06.2022 | Fieguth, Arnold | Randbedingungen |
| | Bohnert | Lösungsstrategie |
| | Arnold, Fieguth, Bohnert | Bausteinsicht |
| | | |
| 10.06.2022 | Fieguth | Laufzeitsicht Diagramm |
| | Arnold | Laufzeitsicht Diagramm |
| | Bohnert | Laufzeitsicht Diagramm |
| | | |
| 11.06.2022 | Fieguth, Arnold, Bohnert | Laufzeitsicht Diagramm ergänzt |
| | | Texte verfasst |
| | | |
| 13.06.2022 | Fieguth, Arnold, Bohnert | Bausteinsicht überarbeitet |
| | | Laufzeitsicht überarbeitet |
| | | |
| 15.06.2022 | Fieguth, Arnold, Bohnert | Bausteinsicht, Laufzeitsicht ergänzt |
| | | |
| | | |
| | | |
| | | |
| | | |

# Inhalt

Versionshistorie ..... 1
Einführung und Ziele ..... 3
Aufgabenstellung ..... 3
Qualitätsziele ..... 3
Usability / Benutzbarkeit ..... 3
Verlässlichkeit, Robustheit ..... 3
Performance, Skalierbarkeit ..... 3
Randbedingungen ..... 4
Lösungsstrategie ..... 4
Bausteinsicht ..... 5
Laufzeitsicht ..... 7

# Einführung und Ziele

## Aufgabenstellung

Entwickelt wird das SPAsS-Tool (Semester-Planungs-Anwendung für studierende Studierende). Hierbei handelt es sich um einen individuellen interaktiven Studienplaner.

Das System ist dazu gedacht Studierenden den Aufbau und die Planung ihres Studiums (Modulwahl etc.) zu erleichtern.

Es wird möglich sein, dass jeder Studierende sich einen individuellen Stundenplan bauen kann – schließlich gibt es individuelle Fälle der Studierenden, die eine Planung abseits der Regelstudienzeit erfordern.

Es ist uns wichtig, die Anwendung Benutzerfreundlich zu gestalten, so dass der User mit gezielten Schritten durch unser System geleitet wird.

## Qualitätsziele

### Usability / Benutzbarkeit

- JavaFX GUI muss so eindeutig gestaltet sein, dass der Benutzer es ohne Erklärung individuell nutzen kann
- Leistungen / Module (Kästchen) können per Drag & Drop zwischen Semestern verschoben werden (Orientierung an bekannten Angebots-Restriktionen)

### Verlässlichkeit, Robustheit

- Mindestens alle Hauptfunktionen müssen anhand von JUnit5-Tests testbar sein
- Der Benutzer muss darauf hingewiesen werden, wenn er falsch mit dem System interagiert – zum Beispiel die Fortschrittsregelung beim Verschieben der Module nicht einhält

### Performance, Skalierbarkeit

- Erweiterbarkeit / Anpassbarkeit, Einlesbarkeit von mehreren Formaten soll möglich sein oder einfach anpassbar
- Jeder Student kann seinen individuellen Studienplan im System einpflegen

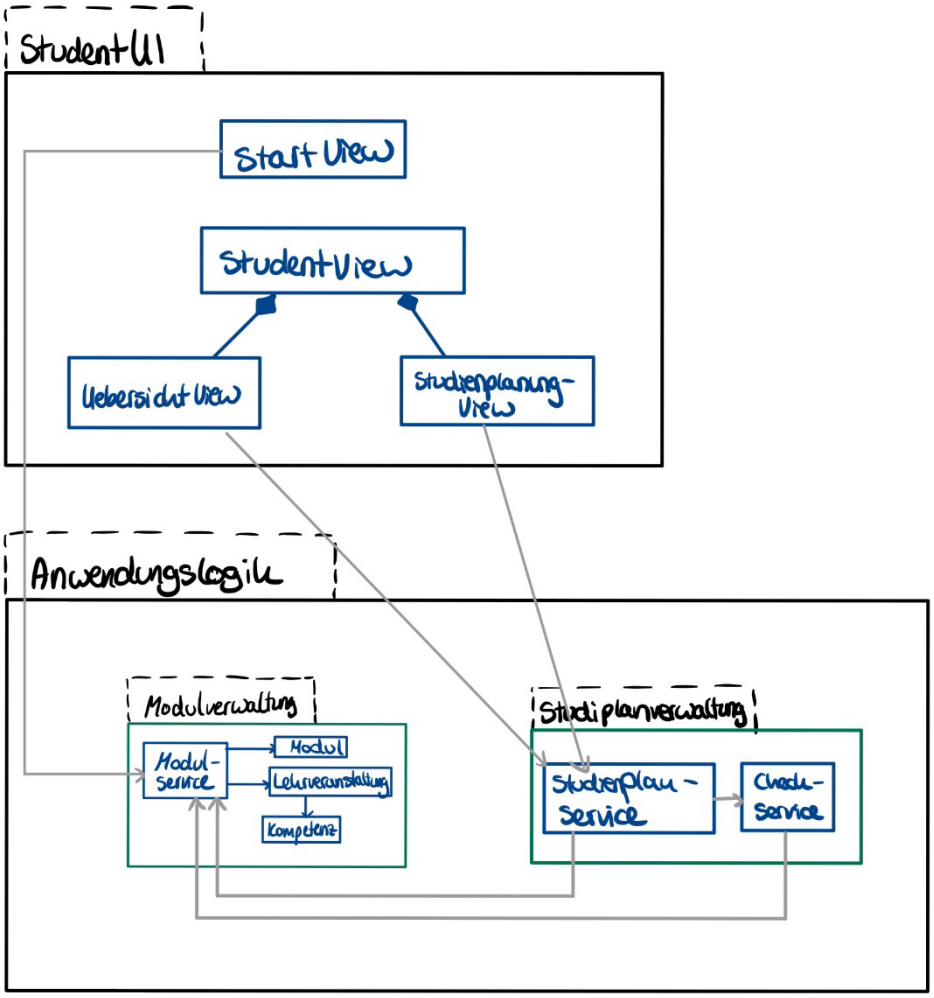
## Randbedingungen

Unser Team besteht aus 3 Leuten (Arthur Fieguth, Beate Arnold und Marie Bohnert).
Der zu erfüllende Zeitraum streckt sich vom 06. Mai 2022 bis 07. Juli 2022.
Es sind zwei Zwischenbesprechungen vereinbart. Am 27. Mai wird der Standpunkt „Anforderungsspezifikation“ (Was soll gemacht werden) überprüft. Am 17. Juni wird das „Design-Dokument“ (Konzept der softwaretechnischen Umsetzung) vorgestellt. Es wird nach Abgabe des Projekts eine Vorstellung der fertigen Anwendung geben.
Die technischen Rahmenbedingungen setzen sich aus einer Linux funktionsfähigen Anwendung mit Gradle build-, test- und ausführbaren JavaFX-Anwendung (Java 17) zusammen.

## Lösungsstrategie

Wir haben uns dazu entschieden unser System mit Hilfe einer Schichtenarchitektur aufzubauen. Dabei werden die zwei Schichten UI und Anwendungslogik unterschieden. Es erfolgt eine Trennung zwischen GUI und eigentlicher Programmlogik, sodass in unserem System die UI leicht austauschbar bleibt.

## Bausteinsicht




Wird das Programm das erste Mal gestarten gelangt der Benutzer auf die StartView. Hier hat er die Möglichkeit sein Curriculum per Dateiauswahl (Button) in die Anwendung zu laden. Bei erfolgreichem Einlesen gelangt er auf die StudentView, welche die ÜbersichtsView und die StudienplanungView beinhaltet.

In der ÜbersichtsView wird dem Studierenden angezeigt, wieviele CP er bereits erreicht hat.

In der StudienplanungView hat der Studierende eine Übersicht über alle Semester mit den jeweiligen Modulen. Jedes Modul wird hier mit Modulname CP Anzahl, aufgesplittet in Prüfung und ggf. Praktikum, sowie ob das Modul im Sommer- und/oder Wintersemester aneboten wird, angezeigt.

Die StudienplanungsView besteht zu Beginn aus sieben Regelstudienzeitsemestern und stellt zwei Zusatzzeilen zur Verfügung. Die Module können per Drag and Drop in die Zusatzsemester verschoben werden, damit der individuelle Plan ausgedehnt werden kann.

Durch einen Plus Button können weitere Zeilen ergänzt werden, falls die Gesamtsemesterzahl größer als neun ist.

Die StudentView hat eine Referenz des StudienplanService, welcher für das Laden und Speichern des individuellen Plans verantwortlich ist.

Die ladePlan Methode holt sich mit Hilfe des ModulListen Getters des ModulService alle Module und setzt die Koordinaten der einzelnen Module. Dies feuert ein Event, woraufhin sich die StudienplanungView anpasst.

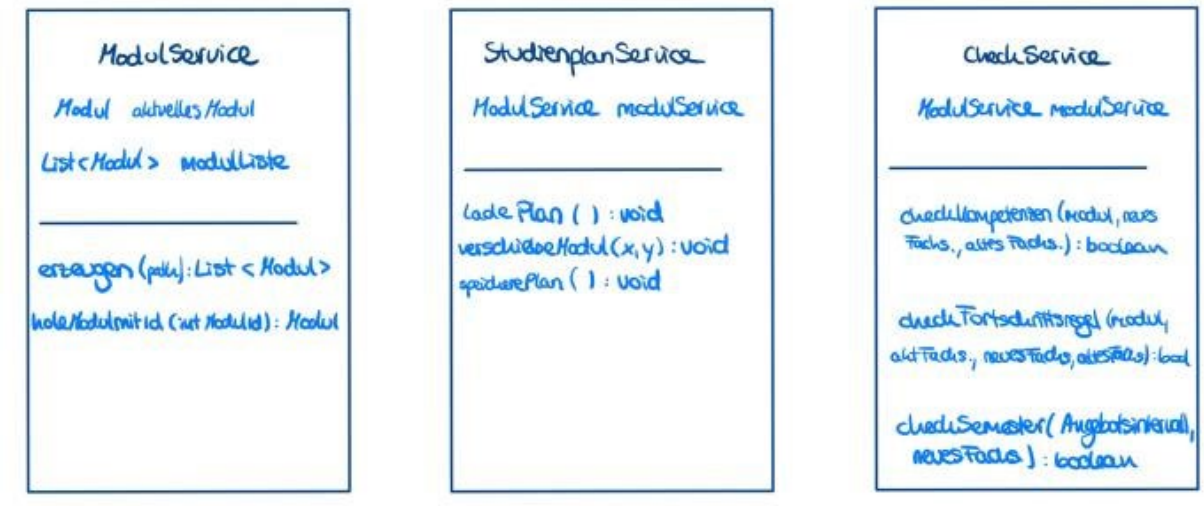
Die verschiebeModul Methode ruft die checkMethoden des CheckServices auf, bevor es ein Modul verschieben kann. Wurde das Modul verschoben, wird ein changeEvent gefeuert, woraufhin sich die StudienplanungView anpasst.

Die speicherePlan Methode erzeugt oder updatet eine XML-Datei („individuellerPlan.xml“), welche die Anordnung der aktuellen Studienplanung speichert.

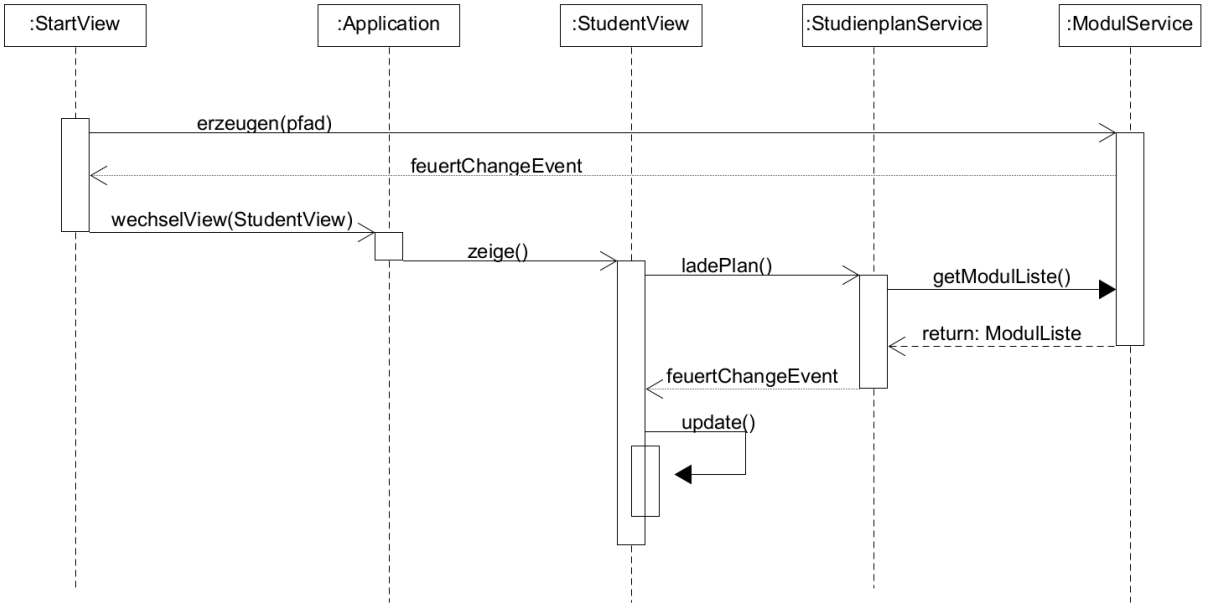
Der StudienplanService hat eine Referenz des CheckService. Dieser ist dafür verantwortlich, das Verschieben eines Moduls zu validieren.

Der StudienplanService und der CheckService sind im Package Studiplanverwaltung enthalten. Beide haben Referenzen auf den ModulService, welcher alle Module verwaltet. Dazu muss er Referenzen der Klassen Modul und Lehrveranstaltung haben und die Klasse Lehrveranstaltung der Klasse Kompetenz.

Zum Persistieren unserer Daten benutzen wir XML-Dateien. Da diese keine Klassen sind, werden sie nicht in der Bausteinsicht dargestellt. Die CurriculumXML beinhaltet den Regelstudienplan und wird zu Beginn geladen. Der veränderte Plan wird in der IndividuellerPlanXML gespeichert und wird bei erneutem Öffnen der Anwendung geladen.

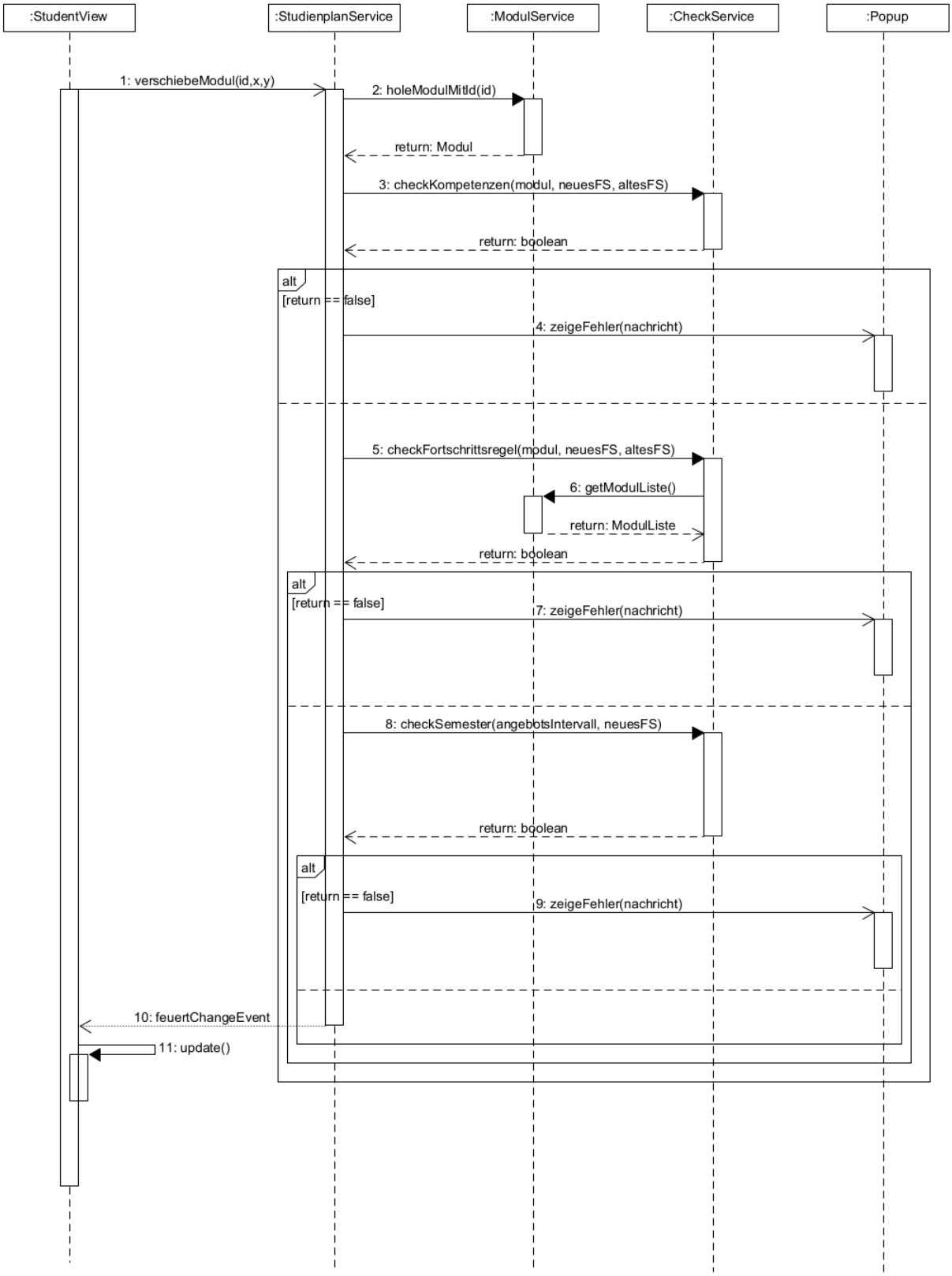


## Laufzeitsicht



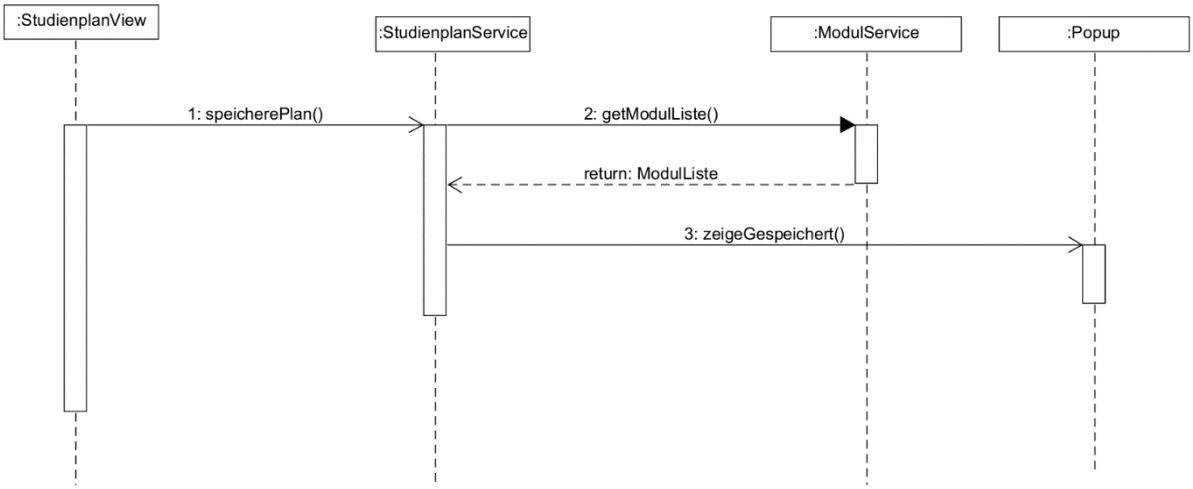

Das Sequenz-Diagramm zeigt den Ablauf, wie man einen Plan in die Anwendung lädt und dieser daraufhin angezeigt wird.

Der Benutzer stößt den Prozess an, indem er auf der StartView per Button-Klick eine Datei, bestehend aus seinem Curriculum, auswählt. Diese Datei wird nun von der View angestoßen, im ModulService durch dessen erzeugen Methode eingelesen. Mit Hilfe des mitgegebenen Dateipfades kann die Methode die Datei Zeile für Zeile einlesen und sämtliche Modulinformationen in einem neuen Modul Objekt speichern. Wurde jedes Modul aus der Datei erfolgreich eingelesen und alle Module in eine Liste gepackt, feuert der ModulService ein PropertyChangeEvent. Die StartView weiß somit, dass der ModulService alle Module eingelesen hat und stößt nun auf der Application die switchView Methode an. Die View wird daraufhin zur StudentView gewechselt. Direkt mit dem Wechsel wird auch die Methode ladePlan des StudienplanService angestoßen. Sie ist dafür verantwortlich allen Modulen, die eingelesen wurden, nun ihre richtigen Pixelkoordinaten zuzuweisen, damit sie nachher richtig von der GUI angezeigt werden können. Dafür holt sich der StudienplanService zunächst vom ModulService die ModulListe mit deren GetterFunktion und setzt anschließend die Positionswerte der einzelnen Module um. Dies feuert erneut ein PropertyChangeEvent, woraufhin die StudentView ihre update Methode aufruft und letztendlich den Plan in der View anzeigt.


:StudentView
:StudienplanService
:ModulService
:CheckService
:Popup
1: verschiebeModul(id,x,y)
2: holeModulMitId(id)
return: Modul
3: checkKompetenzen(modul, neuesFS, altesFS)
return: boolean
alt
[return == false]
4: zeigeFehler(nachricht)
5: checkFortschrittsregel(modul, neuesFS, altesFS)
6: getModulListe()
return: ModulListe
return: boolean
alt
[return == false]
7: zeigeFehler(nachricht)
8: checkSemester(angebotsIntervall, neuesFS)
return: boolean
alt
[return == false]
9: zeigeFehler(nachricht)
10: feuertChangeEvent
11: update()

Das zweite Sequenzdiagramm zeigt den Verschiebe-Vorgang eines Moduls. Der Benutzer stößt den Prozess an, indem er sich per Drag & Drop ein Modul nimmt und versucht es an eine andere Stelle zu verschieben. Dies löst einen Vorgang auf der StudentView aus indem auf Anwendungslogik Ebene die VerschiebeModulMethode des StudienplanService aufgerufen wird. Die Methode bekommt die x und y Koordinaten der zu testenden neuen Position, sowie die id des zu verschiebenden Moduls mitgegeben. Um das Verschieben umsetzen zu können, holt sich der StudienplanService zunächst das Modul vom ModulService mit der holeModulmitId Methode. Nach erfolgreichem return des Moduls wird nun gecheckt, ob eine Verschiebung möglich ist. Hierzu werden mehrere check-Methoden des checkService nacheinander aufgerufen. Der Vorgang beginnt mit der checkKompetenzen Methode. Diese checkt, ob alle erforderlichen Kompetenzen erfüllt sind, um das Modul im Zielfachsemester zu absolvieren. Ist dies nicht der Fall, bleibt das Modul an Ort und Stelle und ein Fehlerhinweis wird dem Benutzer per Popup angezeigt. Wurden alle Kompetenzen erfüllt, wird die Fortschrittsregelung überprüft. Hierzu werden alle Semester durchgegangen und geschaut, welche Module in welchen Semestern bereits bestanden sind. Dafür benötigt der checkService die Modulliste des ModulService. Auch hier wird bei einem Fehler ein Popup angezeigt. Als letztes wird die checkSemesterMethode ausgeführt. Diese überprüft, ob das Modul im Zielsemester überhaupt angeboten wird. Ob der Benutzer richtig liegt, sieht er, wenn sich das Modul verschiebt 😉

Wurden alle Fälle geprüft, wird ein changeEvent gefeuert. Die View ruft daraufhin ihre update Methode auf und zeigt die Verschiebung des Moduls auf der GUI an.




Das dritte Sequenzdiagramm zeigt den Ablauf des Speicherns eines individuellen Plans. Der Benutzer stößt den Prozess an, indem er auf der StudienplanView auf den speichernButton klickt. Dadurch wird die speicherePlan Methode des StudienplanService aufgerufen. Die speicherPlan Methode benötigt die individuellen Anordnungen der Module, welche er sich mithilfe eines getters aus der ModulPlanService besorgt. Hat er diese erhalten, schreibt er alle Daten in eine XML Datei, damit bei erneutem aufrufen des Programms die Daten wieder geladen werden können. Anschließend öffnet sich ein Popup, das den Erfolg des speicherns kommuniziert.