Dokumentation - MindMap

Dokumentation für das Modul Projekt 1

Studiengang:	[Informatik]
Autoren:	[Dominik Meister & Lorenz Rasch]
Betreuer:	[Prof. Dr. Olivier Biberstein]
Datum:	12.06.2019

Inhaltsverzeichnis

1 Vision **1**

2 Analysis **3**
2.1 Domainmodel . . . 3
2.2 Use Cases . . . 4
2.3 User Stories . . . 8
2.4 Systemkontext . . . 8
2.5 Anforderungen . . . 9

3 Design **11**
3.1 System Sequence Diagram . . . 11
3.2 Sequence Diagram . . . 12
3.3 Package Diagramm . . . 17
3.4 UML Diagramme . . . 18

4 Entwicklung **19**
4.1 Sprint Planung . . . 19
4.2 Sprint 1 . . . 19
4.3 Sprint 2 . . . 20
4.4 Sprint 3 . . . 20
4.5 Sprint 4 . . . 20
4.6 Kontroller Management . . . 20
4.7 Ordnungs Funktion . . . 20
4.8 Programm ausführen . . . 21

5 Fazit **23**

Abbildungsverzeichnis **25**

Versionen

Version	Datum	Status	Bemerkungen
0.1	13.03.2019	Entwurf	Beginn Projekt
0.2	02.04.2019	Entwurf	Bericht nach BFH Vorlage
1.0	12.06.2019	Definitiv	Version für Abgabe des Projekts

1 Vision

Jeder kennt diese Situation, man sitzt in einem Meeting und möchte kurz ein Brainstorming machen. Oder man möchte für die nächste Arbeit zu einem Thema eine Übersicht zu den Informationen erstellen. Eine der einfachsten Methoden so eine Übersicht darzustellen ist das Mindmap. Das Mindmap auch Gedankenlandkarte genannt ist eine Technik welche von Tony Buzan geprägt und entwickelt wurde. Das Mindmap basiert auf dem Prinzip der Assoziation. Dies kommt nicht von ungefähr, unser Gehirn arbeitet ebenfalls mit Assoziationen, es versucht ständig neue Informationen mit gewissen Kategorien und anderen Informationen zu verknüpfen. Das Mindmap basiert auf derselben Technik, deshalb fällt es uns auch sehr einfach ein Mindmap zu erstellen. Dieses zu erstellen ist jedoch mit den meisten Programmen eher mühsam, deshalb greift man auf den Stift und Papier zurück. Um hier Abhilfe zu schaffen kommt unser Projekt ins Spiel.

Ziel unseres Projektes, ist das Entwickeln einer Software mit welcher man möglichst einfach und schnell ein Mindmap erstellen kann. Dabei werden wir besonders Wert auf die Benutzerfreundlichkeit legen. Es sollte möglich sein in kürzester Zeit ein Mindmap zu erstellen. Das Programm sollte aber auch reif sein für komplexere Mindmaps. Deshalb wird das Programm auch Funktionen wie verschiedene Verbindungstypen und Farben unterstützen, um auch komplexere Mindmaps übersichtlich zu gestalten. Wichtige Funktionen welche das Programm ebenfalls bieten muss sind: das Speichern, Laden und Drucken/Exportieren der Mindmaps.
Das Programm wird als Standalone Software in JavaFX erstellt. JavaFX ist ein Framework von Oracle welches auf die Erstellung von GUI's und Multimedialen Inhalten spezialisiert ist.

Wir hoffen wir können durch dieses Projekt vielen Menschen helfen Ihre Ideen mithilfe unseres Programmes festzuhalten.

2 Analysis

In diesem Abschnitt wird eine grobe Übersicht über die Software erstellt. Fragen wie, was sind die Anforderungen an die Software, was sind die Anforderungen der Akteure etc. werden hier beantwortet. Ebenfalls wird eine grobe Übersicht über die Programmstruktur erstellt.

2.1 Domainmodel

In diesem Abschnitt wird das Domain Model ("Fachmodell") des Mindmap Programmes beschrieben. Es stellt die wichtigsten Fachklassen und Assoziationen dar und zeigt die jeweiligen Multiplizitäten.

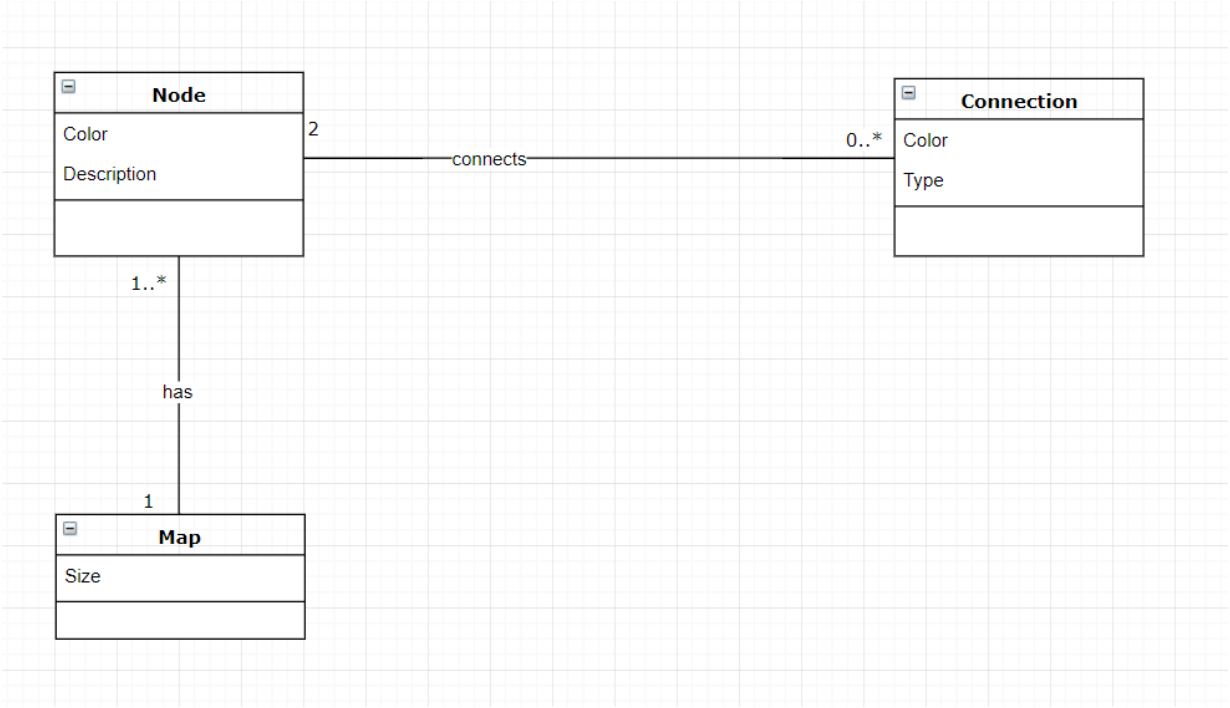

Abbildung 2.1: Domainmodel

Wie oben in der Grafik zu sehen ist, haben wir uns dafür entschieden, dass eine Map mindestens einen Knoten haben muss um eine Map zu sein. Im Moment denken wir dass die Connection und die Map keine direkte Verbindung benötigen. Dies könnte sich aber beim Implementieren vielleicht noch ändern. Ebenfalls haben wird definiert, dass eine Verbindung immer zwischen 2 Knoten besteht.

2.1.1 Assoziationen

- Zwei Knoten haben keine oder mehrere Verbindungen. Eine Verbindung besteht aus 2 Knoten.
- Eine Map hat einen oder mehrere Knoten. Ein Knoten gehört zu einer Map.

2.2 Use Cases

Use-Cases beschreiben die Requirements der einzelnen Akteure und zeigen die Abläufe des Programmes auf. Dabei wird aber nicht auf die explizite Funktion der einzelnen Teile eingegangen.

2.2.1 Use Case Diagramm

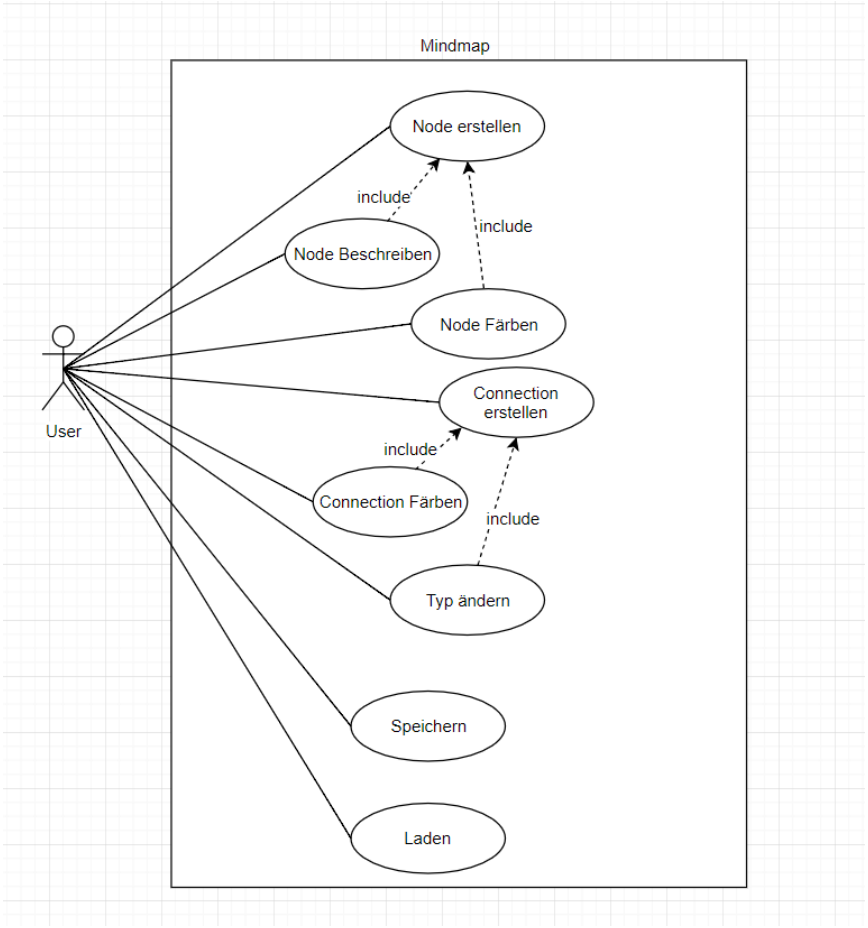

Abbildung 2.2: Use Case Diagramm

2.2.2 Use Cases

Use Case 1: Knoten erstellen

- **Titel:** Als Benutzer möchte ich einen Knoten erstellen können.
- **Voraussetzung:** Das Programm ist gestartet.
- **Erfolgsszenario:**
  1. Der Benutzer gibt den Befehl einen neuen Knoten zu erstellen.
  2. Das Programm fügt den neuen Knoten dem Mindmap hinzu.
- **Nachbedingung:** Ein neuer Knoten wurde dem Mindmap hinzugefügt.
- **Alternative Szenarien:**

1.a 1 Der Benutzer schreibt eine Beschreibung für den Knoten.

1.b 1 Der Benutzer ändert die Farbe des Knoten.

Use Case 2: Knoten verbinden

- **Titel:** Als Benutzer möchte ich zwei Knoten miteinander verbinden können.
- **Voraussetzung:** Ein Mindmap mit mindestens zwei Knoten wurde erstellt oder geladen.
- **Erfolgsszenario:**
    1. Der Benutzer wählt zwei Knoten aus.
    2. Das Programm verbindet die ausgewählten Knoten.
- **Nachbedingung:** Zwischen den ausgewählten Knoten besteht eine Verbindung.
- **Alternative Szenarien:**

1.a 1 Der Benutzer ändert die Art der Verbindung.

Use Case 3: Mindmap speichern

- **Titel:** Als Benutzer möchte ich ein Mindmap speichern können.
- **Voraussetzung:** Ein Mindmap wurde erstellt.
- **Erfolgsszenario:**
    1. Der Benutzer wählt einen Speicherort.
    2. Das Programm übergibt dem Filesystem eine Datei.
- **Nachbedingung:** Eine Datei mit den Informationen des Mindmaps wurde am gewählten Ort im Filesystem gespeichert.
- **Alternative Szenarien:**

2.a 1 Das Filesystem verweigert das Speichern wegen zu wenig Speicherplatz.

2.a 2 Das Programm informiert den Benutzer, dass nicht genug Speicherplatz vorhanden ist.

2.b 1 Das Filesystem verweigert den Zugriff auf den Speicherort.

2.b 2 Das Programm informiert den Benutzer, dass der Zugriff verweigert wurde.

Use Case 4: Mindmap laden

- **Titel:** Als Benutzer möchte ich ein gespeichertes Mindmap laden können.
- **Voraussetzung:** Eine Datei mit den Informationen eines Mindmap existiert im Filesystem.
- **Erfolgsszenario:**
    1. Der Benutzer wählt eine Datei aus.
    2. Das Programm lädt die Datei.
    3. Das Programm zeigt das geladene Mindmap an.
- **Nachbedingung:** Das Programm zeigt ein Mindmap mit den Informationen aus der geladenen Datei.
- **Alternative Szenarien:**

2.a 1 Das Filesystem verweigert den Zugriff auf die Datei.

2.a 2 Das Programm informiert den Benutzer, dass der Zugriff verweigert wurde.

2.b 1 Das Programm lädt eine beschädigte Datei.

2.b 2 Das Programm informiert den Benutzer, dass die Datei beschädigt ist und nicht geladen werden kann.

Use Case 5: Knoten verändern

- **Titel:** Als Benutzer möchte ich einen erstellten Knoten verändern können.
- **Voraussetzung:** Ein Mindmap mit mindestens einem Knoten wurde erstellt oder geladen.
- **Erfolgsszenario:**
  1. Der Benutzer wählt einen existierenden Knoten.
  2. Der Benutzer schreibt eine (neue) Beschreibung für den Knoten.
- **Nachbedingung:** Der gewählte Knoten wurde verändert.
- **Alternative Szenarien:**

2.a 1 Der Benutzer wählt eine (neue) Farbe für den Knoten.

Use Case 6: Verbindung verändern

- **Titel:** Als Benutzer möchte ich verschiedene Verbindungstypen definieren können.
- **Voraussetzung:** Ein Mindmap mit mindestens zwei Knoten wurde erstellt oder geladen.
- **Erfolgsszenario:**
  1. Der Benutzer wählt eine Verbindung aus.
  2. Der Benutzer ändert die Art/die Darstellung der Verbindung.
- **Nachbedingung:** Die gewählte Verbindung wurde verändert.

Use Case 7: Mindmap drucken

- **Titel:** Als Benutzer möchte ich ein Mindmap drucken können.
- **Voraussetzung:** Ein Mindmap wurde erstellt oder geladen.
- **Erfolgsszenario:**
  1. Der Benutzer gibt den Befehl das Mindmap zu drucken.
  2. Das Programm sendet eine Datei an den Drucker.
- **Nachbedingung:** Das Mindmap wurde ausgedruckt.
- **Alternative Szenarien:**

2.a 1 Das Programm informiert den Benutzer, dass kein Drucker angeschlossen ist.

2.b 1 Das Programm informiert den Benutzer, dass die gesendete Datei vom Drucker zurückgewiesen wurde.

Use Case 8: Mindmap exportieren

- **Titel:** Als Benutzer möchte ich ein Mindmap exportieren können.
- **Voraussetzung:** Ein Mindmap wurde erstellt oder geladen.
- **Erfolgsszenario:**
  1. Der Benutzer wählt ein Dateiformat.
  2. Der Benutzer wählt einen Speicherort.
  3. Das Programm übergibt dem Filesystem eine Datei mit dem gewünschten Format.
- **Nachbedingung:**
- **Alternative Szenarien:**

2.a 1 Das Filesystem verweigert das Speichern wegen zu wenig Speicherplatz.

2.a 2 Das Programm informiert den Benutzer, dass nicht genug Speicherplatz vorhanden ist.

2.b 1 Das Filesystem verweigert den Zugriff auf den Speicherort.

2.b 2 Das Programm informiert den Benutzer, dass der Zugriff verweigert wurde.

Use Case 9: Darstellung optimieren

- **Titel:** Als Benutzer möchte ich die Darstellung des Mindmaps optimieren können.
- **Voraussetzung:** Ein Mindmap wurde erstellt oder geladen.
- **Erfolgsszenario:**
  1. Der Benutzer gibt den Befehl das Mindmap zu optimieren.
  2. Das Programm optimiert das Mindmap.
- **Nachbedingung:** Ein Mindmap mit möglichst wenig sich überschneidenden Verbindungen.

2.3 User Stories

Eine User-Story beschreibt in ein oder zwei Sätzen eine gewünschte Funktion des Programmes.

1. Als Benutzer möchte ich, dass das Programm selbsterklärend und einfach zu bedienen ist.
2. Als Benutzer möchte ich ein Programm welches schnell startet und sich nicht langsam anfühlt.
3. Als Benutzer möchte ich ein optisch Ansprechendes Design. Als Benutzer finde ich ein Programm besser wenn sein Design State of the Art ist.
4. Als Benutzer möchte ich, dass das Mindmap auf dem Papier gleich aussieht wie es im Programm ausgesehen hat.
5. Als Benutzer möchte ich nicht einen Experten konsultieren müssen um ein so triviales Programm zu installieren.
6. Als Benutzer möchte ich mit dem Programm neue Mindmaps erstellen können.
7. Als Benutzer möchte ich neue Knoten hinzufügen können, welcher eine Beschreibung besitzt und wenn gewünscht auch eine spezielle Farbe haben kann.
8. Als Benutzer möchte ich die verschiedenen Informationen, also Knoten miteinander vernetzen und so gruppieren können.

9. Als Benutzer möchte ich meine Arbeit speichern können und gespeicherte Projekte auch wieder laden können
10. Als Benutzer möchte ich die Beschreibung eines Knoten auch nach seiner Erstellung bearbeiten können.
11. Als Benutzer möchte ich die Farbe eines Knotens nach seiner Erstellung bearbeiten können.
12. Als Benutzer möchte ich verschiedene Verbindungstypen definieren können, ich möchte Verbindungen zwischen verschiedenen Themen speziell hervorheben können.
13. Als Benutzer möchte ich mein Mindmap drucken können.
14. Als Benutzer möchte ich mein Mindmap exportieren können, um es zum Beispiel als Bilddatei in einem Worddokument einfügen zu können.
15. Als Benutzer möchte ich ein Mindmap mit möglichst wenigen überschneidenden Verbindungen, das Programm sollte eine Möglichkeit bieten dieses Problem zu lösen.

2.4 Systemkontext

Der Systemkontext ist eine Beschreibung des Systems, seiner Teile und äusserlichen Einwirkungen auf das System.

Der Benutzer kann mit unserer Software ein Mindmap erstellen. Dies geschieht über eine grafische Benutzeroberfläche. Mit Hilfe eines Algorithmus können die Themen und Verbindungen des Mindmap optimal verteilt werden. Die Benutzeroberfläche und der Algorithmus sind Teil der Software
Die Software greift auf das Dateisystem des Computers zu um ein Mindmap zu speichern oder zu laden. Der Druckservice des Computers erlaubt das Drucken des Mindmaps. Diese beiden Teile liegen ausserhalb unserer Software.

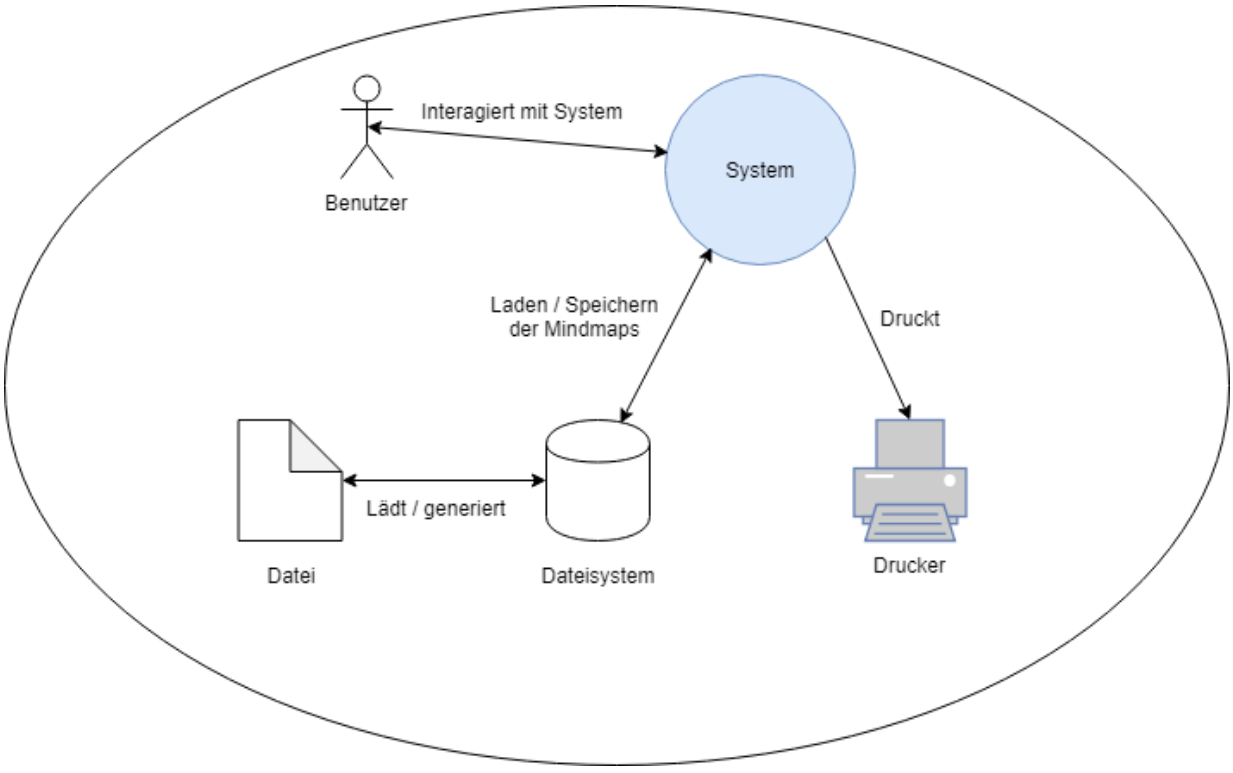

Abbildung 2.3: Systemkontext

3 Design

3.1 System Sequence Diagram

Die System Sequence Diagramme zeigen die möglichen Interaktionen von extern mit unserem Mindmap Programm.

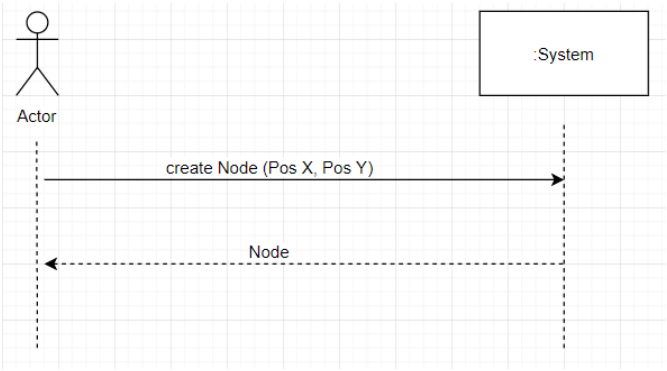

Abbildung 3.1: Create Node SSD

Der Benutzer übergibt dem System eine Position in welcher er den Node erstellen möchte, das System erstellt dann den Node mit allen seinen Komponenten, also der Form, dem Text und den Anchors für das Verbinden.

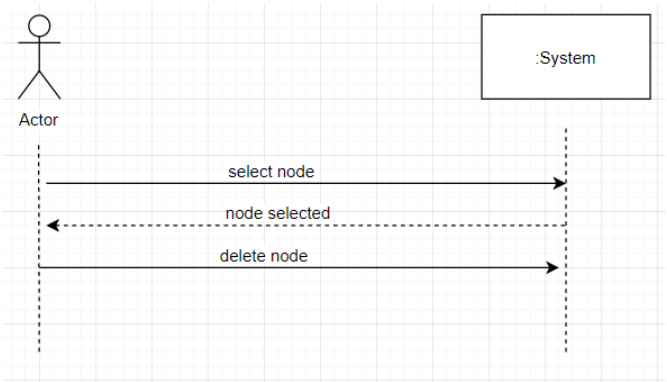

Abbildung 3.2: Delete Node SSD

Der Benutzer wählt einen Node oder eine Verbindung(der Ablauf zum Löschen eines Nodes oder einer Verbindung ist gleich), das System löscht diesen Node dann vom GUI und von der Liste.

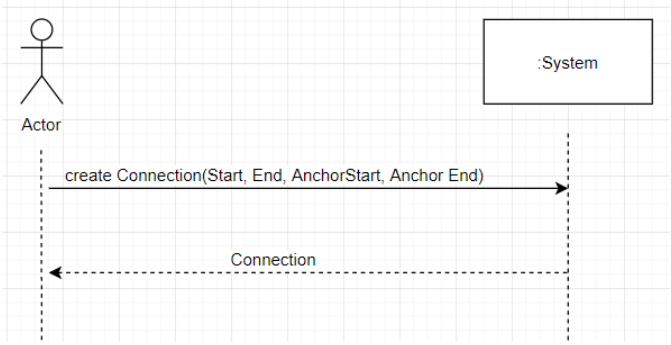

Abbildung 3.3: Create Connection SSD

Der Benutzer wählt einen Start und einen Ziel Anchor für seine Verbindung aus, das System erstellt dann anhand dieser Informationen die Verbindung.

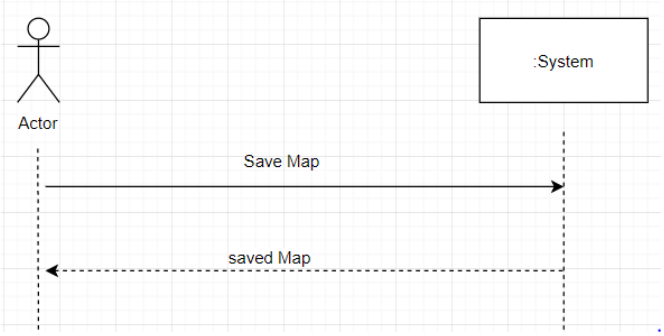

Abbildung 3.4: Save Map SSD

Der Benutzer speichert die Map, das System speichert nun alle Informationen der Map in eine Datei und übergibt sie dem Benutzer.

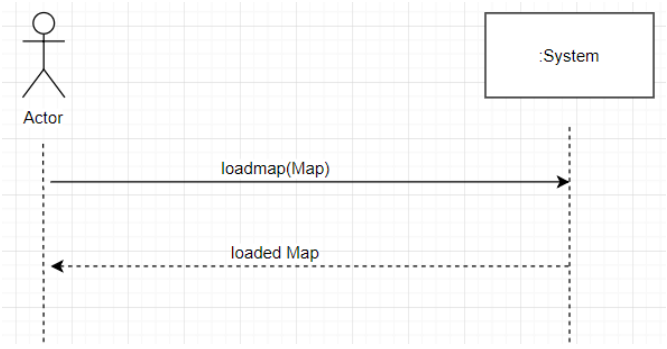

Abbildung 3.5: Load Map SSD

Der Benutzer lädt eine alte Map und übergibt dem System eine Datei mit den Informationen, das System arbeitet nun die Informationen ab und stellt die Map dar.

3.2 Sequence Diagram

Das Sequence Diagramm zeigt die Interaktion zwischen den verschiedenen Objekten in dem Mindmap System.

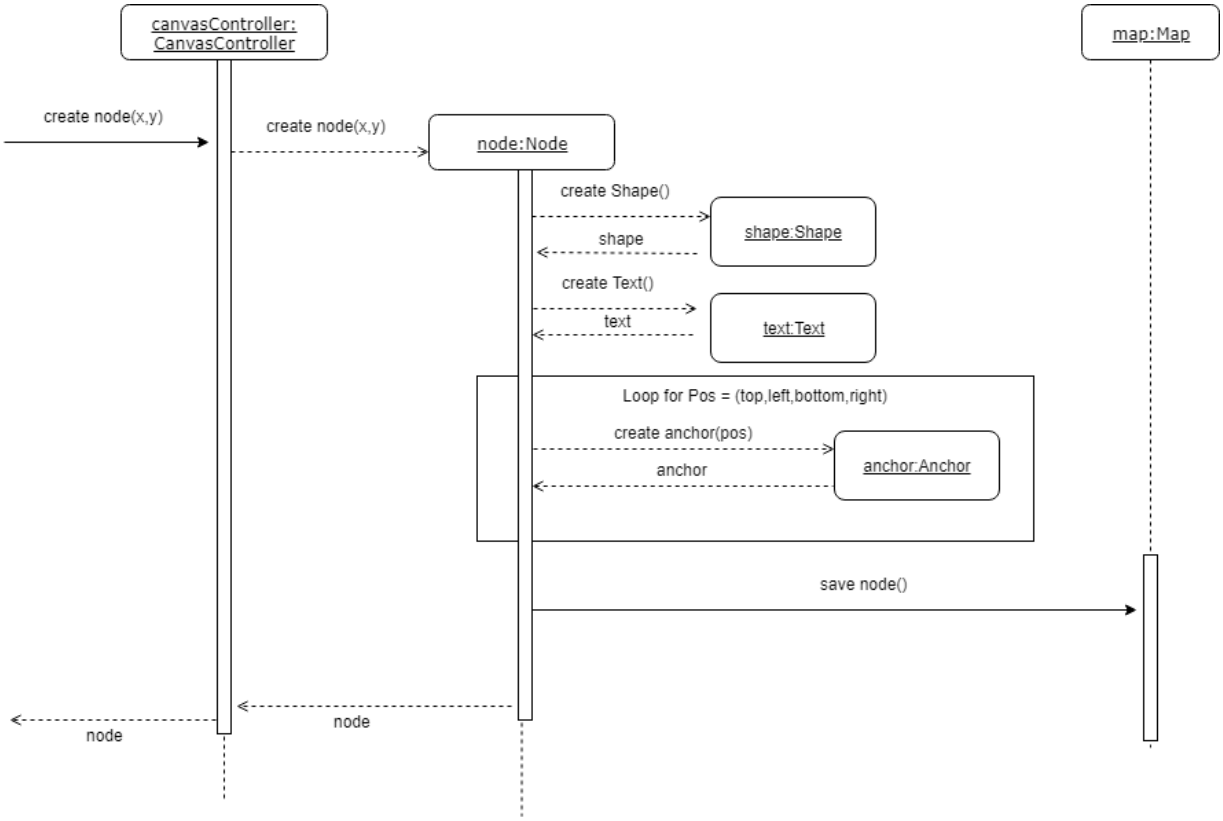

Abbildung 3.6: Create Node SD

Das System erstellt einen neuen Node, der Node erstellt dann seine Objekte welche er benötigt. Er erstellt ein Shape für seine Form, er erstellt einen Text für die Beschreibung und er erstellt 4 Anchors(TOP,LEFT,BOTTOM,RIGHT) für das Verbinden der Knoten. Danach wird der Knoten in die Maplist gespeichert.

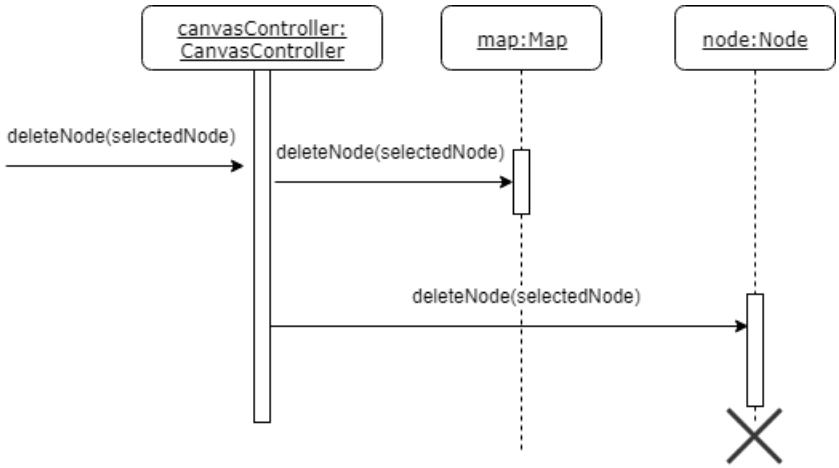

Abbildung 3.7: Delete Node SD

Das System löscht den selektieren Knoten von der Map und löscht diesen dann von der Liste. Dies funktioniert gleich bei der Verbindung.

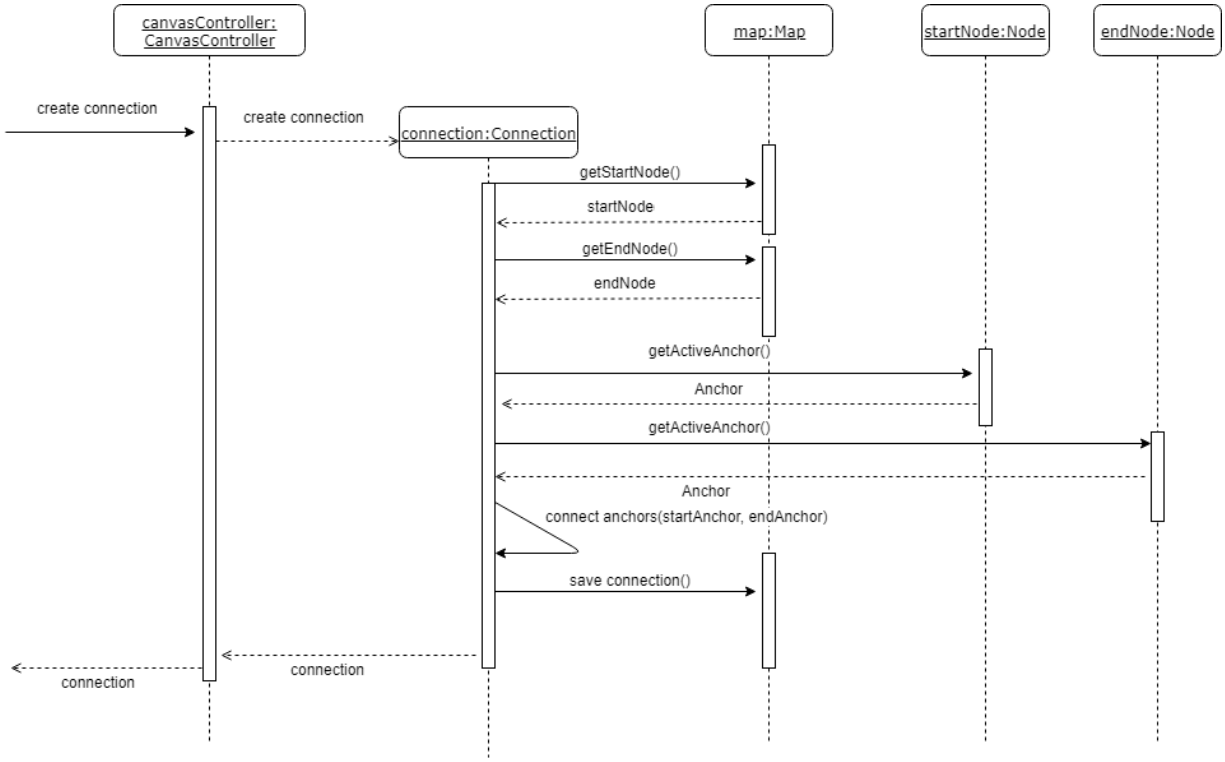

Abbildung 3.8: Create Connection SD

Das System erstellt eine neue Verbindung, es holt sich zuerst die beiden aktiven Knoten, den Start und den End Knoten. Von diesen beiden Knoten holt es sich dann die jeweiligen aktiven Anchors. Danach verbindet das System die beiden aktiven Anchors miteinander.

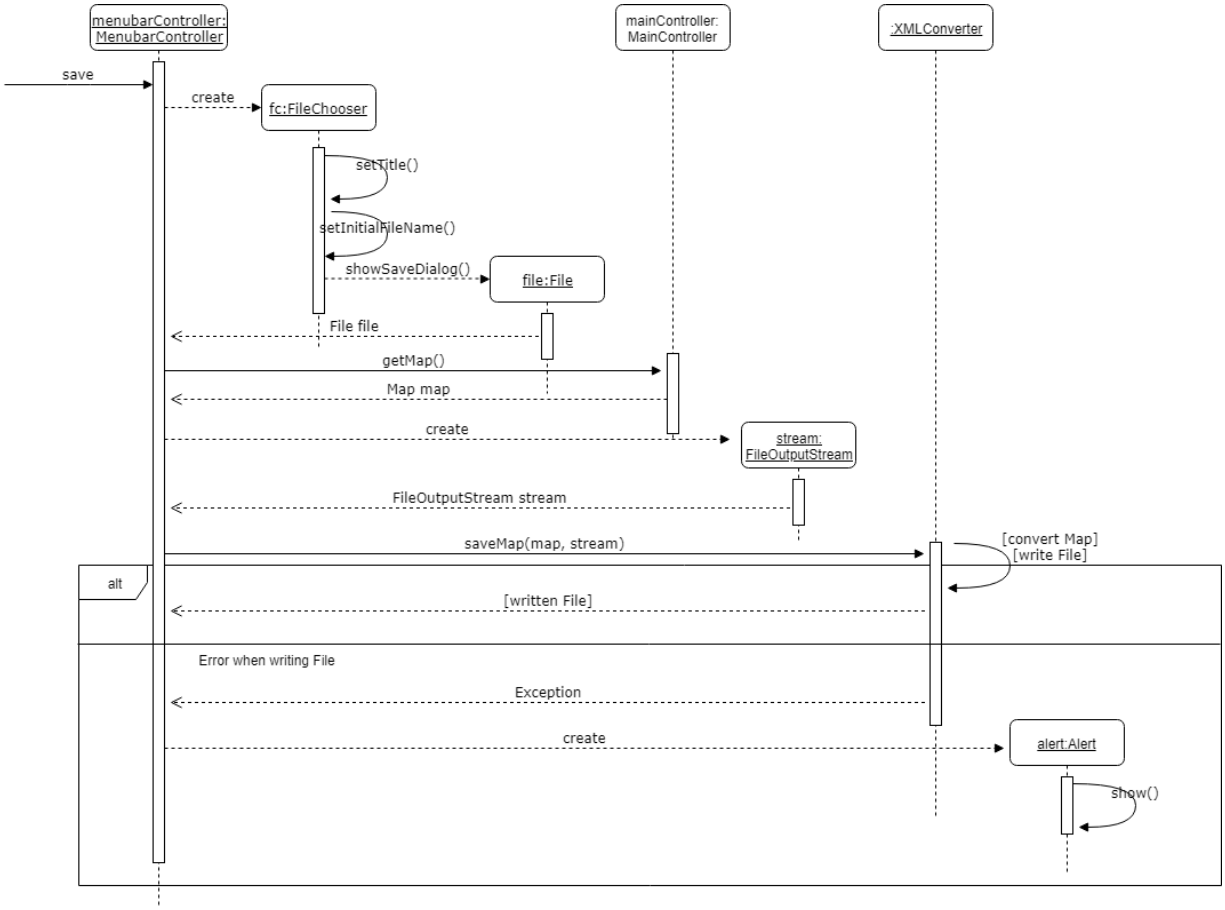

Abbildung 3.9: Save Map SD

Das System lässt den Benutzer eine Datei wählen und erzeugt einen OutputStream. Die Map des MainControllers und der Stream werden dem XMLConverter übergeben. Dieser konvertiert die Map in ein speicherbares Objekt und schreibt die XML Datei. Die Funktionalität des XMLConverters ist im nächsten Sequence Diagram ersichtlich.

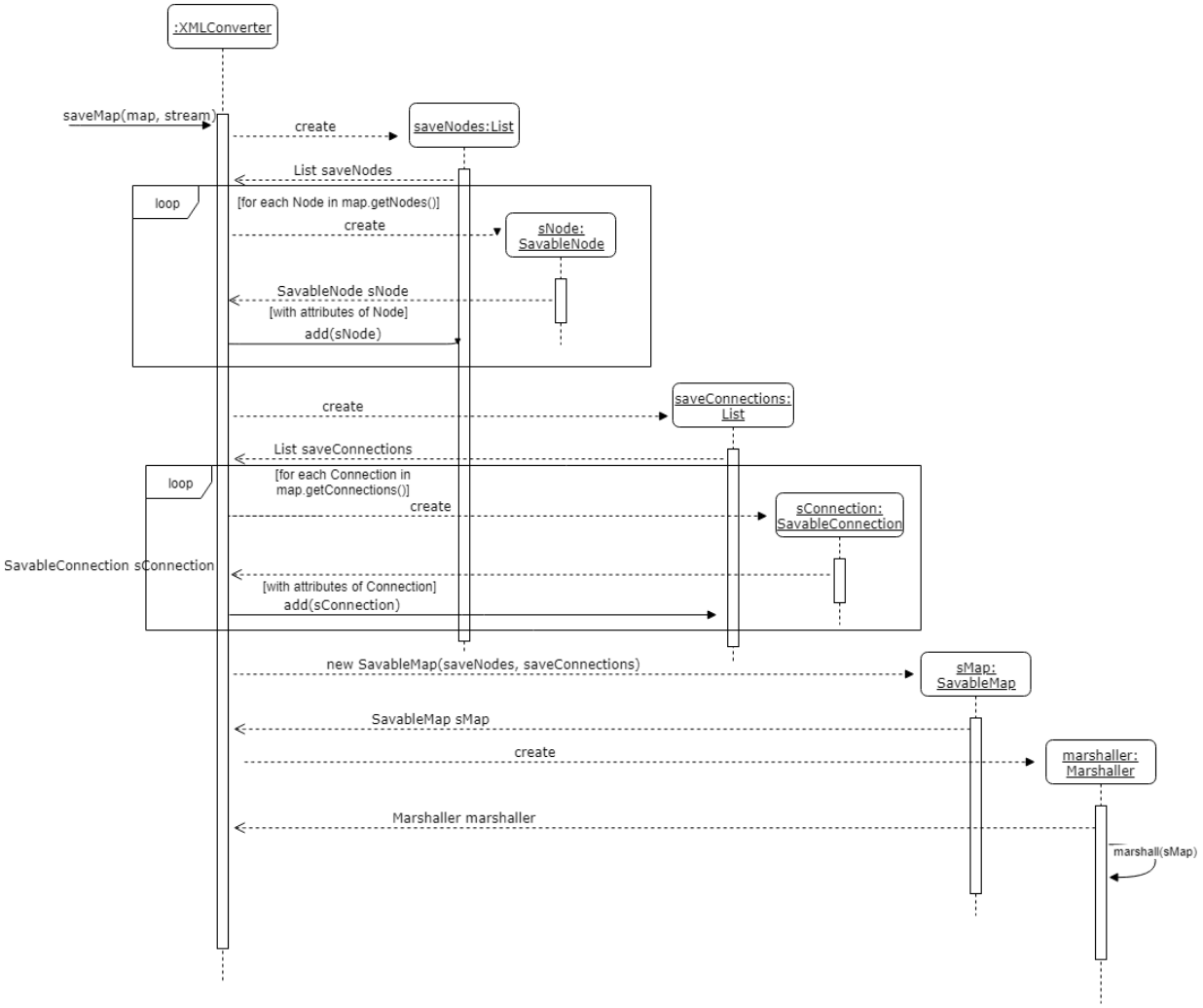

Abbildung 3.10: Convert & Save Map SD

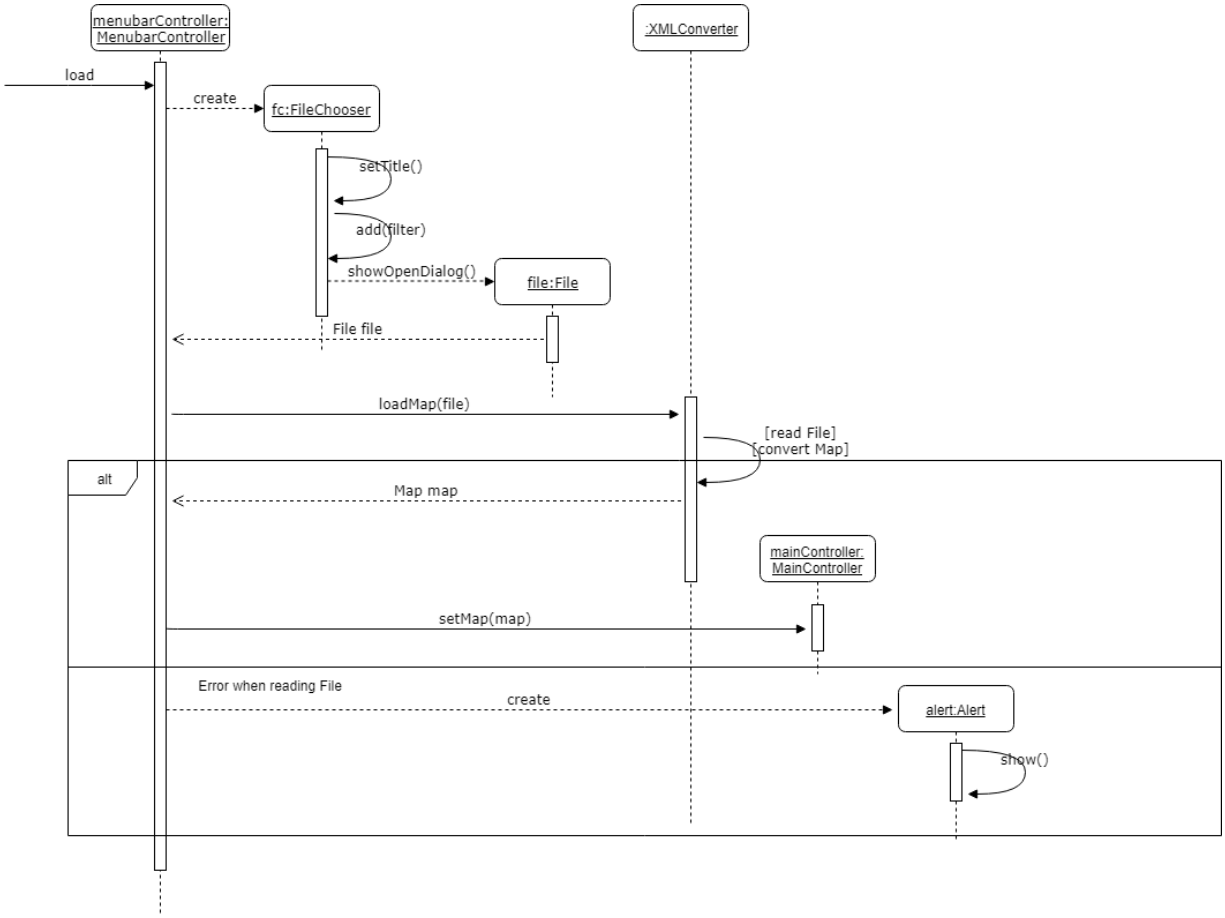

Abbildung 3.11: Load Map SD

Das System lässt den Benutzer eine Datei wählen. Der XMLConverter liest die Datei und erzeugt daraus eine Map. Diese wird danach angezeigt.

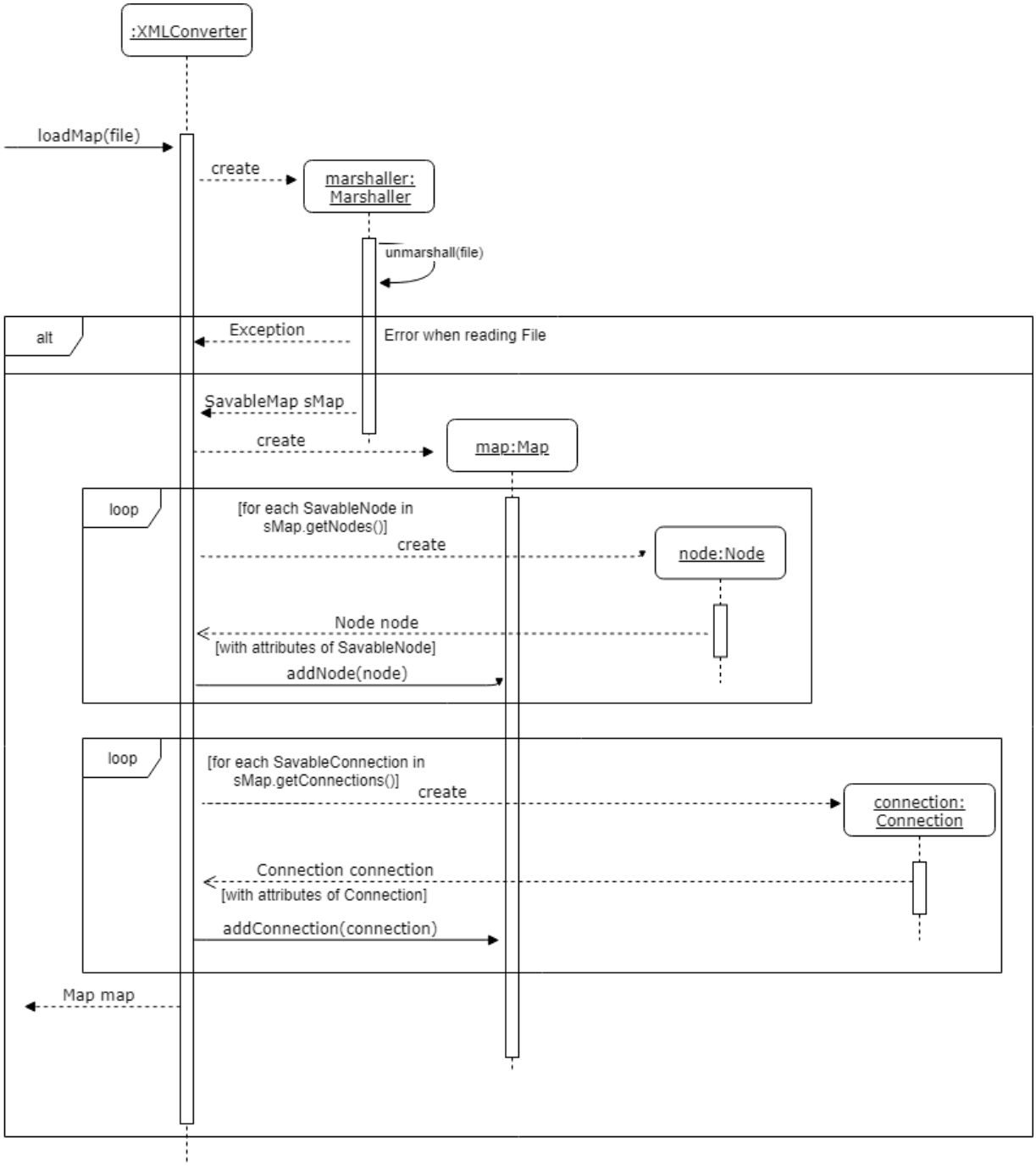

Abbildung 3.12: Load & Convert Map SD

3.3 Package Diagramm

Unsere Packetstruktur ist nach dem MVC Prinzip erstellt. Wir haben ein View Packet mit den FXML Dateien und dem Main Programm, ein Controller Packet mit den Controllern, ein Util Packet mit den Zusatzfunktionen wie Speichern, Laden und Ordnen und ein Model Packet mit den Objekten wie Node, Connection usw.

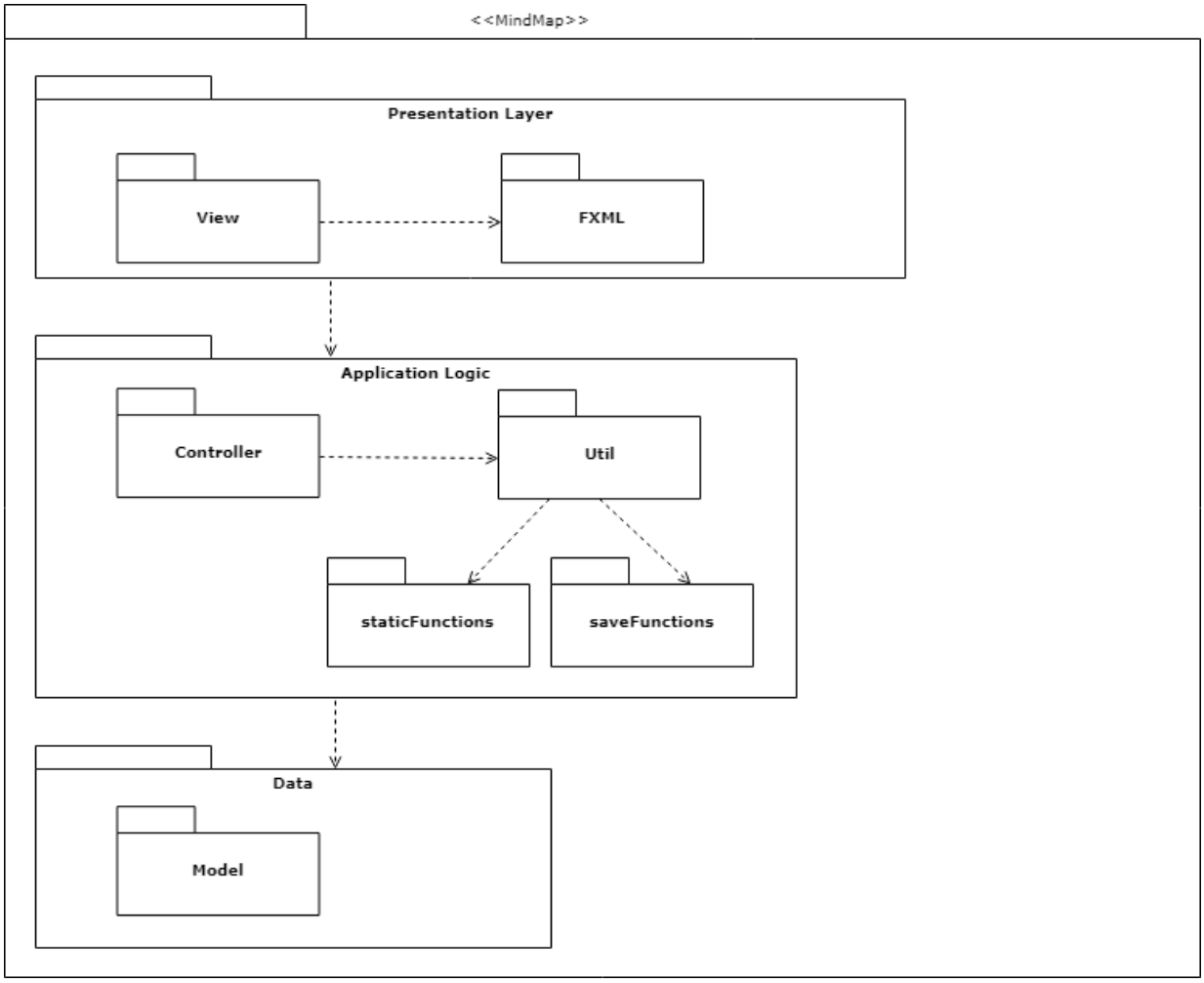

Abbildung 3.13: Package Diagramm

3.4 UML Diagramme

3.4.1 Klassen Diagram

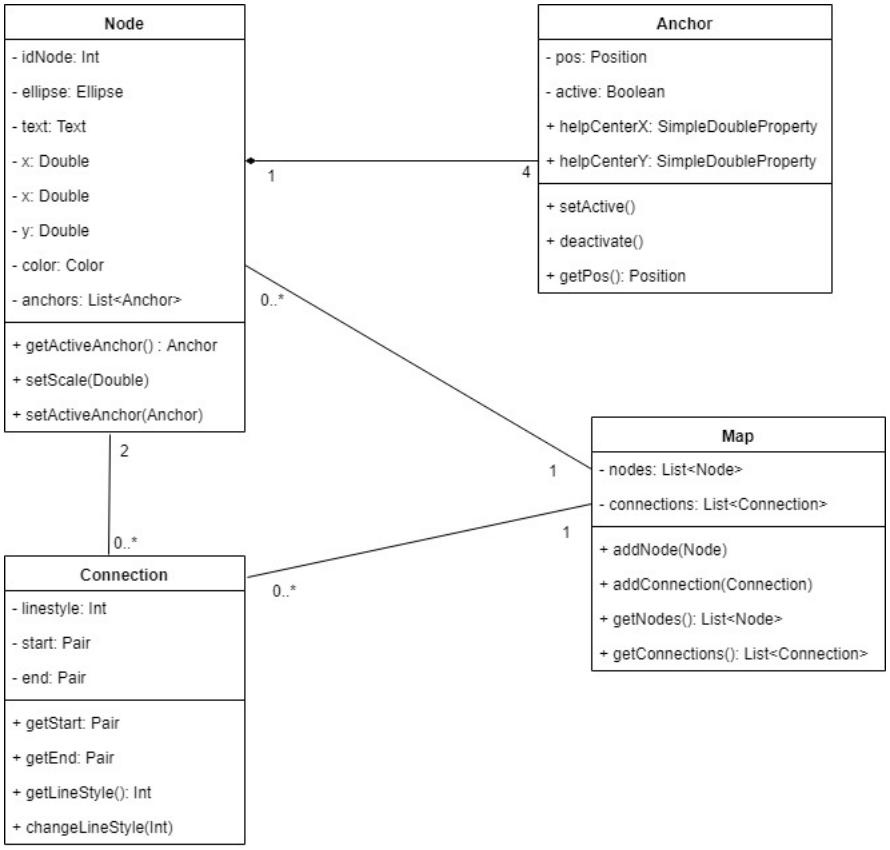

Abbildung 3.14: Klassen Diagramm

Das Klassen Diagramm zeigt einige der wichtigen Klassen in unserem Programm, deren Eigenschaften und wichtige Funktionen.

- **Map**: Eine Map hat eine Liste von Nodes und eine Liste von Connections. Entsprechend können diese ausgelesen oder neue Einträge in den Listen gemacht werden.
- **Node**: Eine Node enthält die vom Programm gezeigte Ellipse und die dazugehörige Position sowie den Text.
- **Connection**: Eine Connection verbindet zwei Nodes (Start und Ende) als Linie.
- **Anchor**: Eine Node hat 4 Anchors an denen eine Linie beginnen kann.

4 Entwicklung

In diesem Kapitel wird die Vorgehensweise und die Realisierung der Entwicklung dokumentiert.

4.1 Sprint Planung

Wir haben uns entschieden das Projekt in Sprints zu unterteilen. Diese Art der Planung erschien uns am praktischsten, da wir so einen guten Überblick hatten wann was fällig ist. Die Sprintplanung wurde im Gitlab integrierten Board gemacht. Zu jeder Userstory wurde ein Issue erstellt und auf die jeweiligen Sprints verteilt.

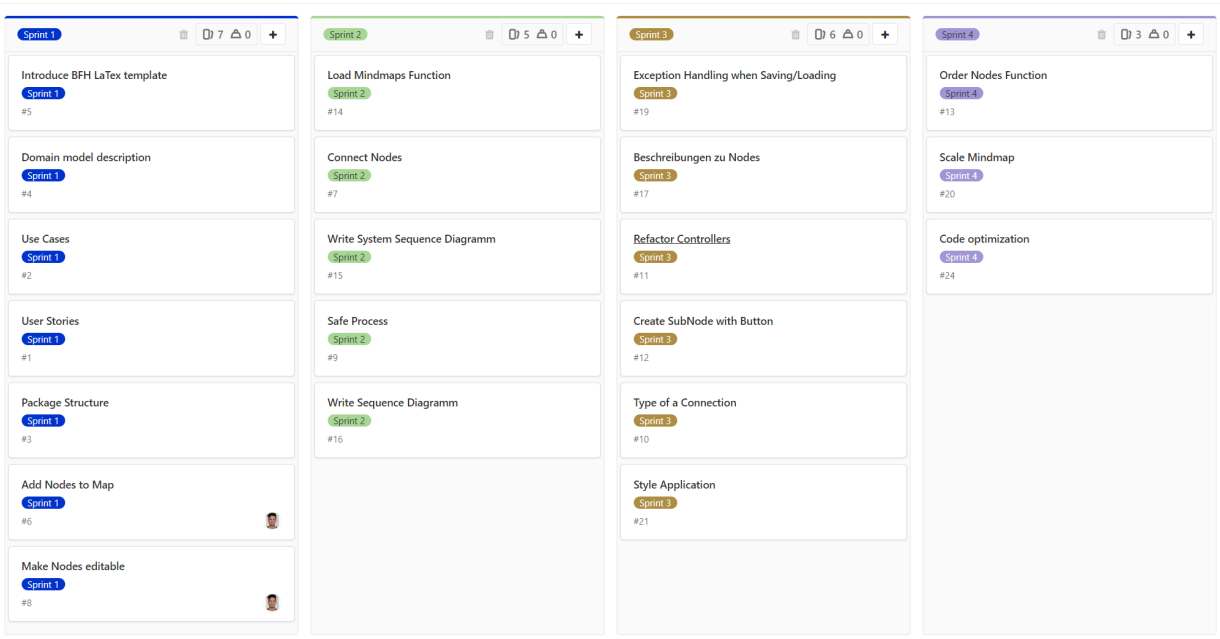

Abbildung 4.1: Sprint Plannung

4.2 Sprint 1

Der Sprint 1 dauerte vom 13.03 - 03.04. In diesem Sprint haben wir primär den Fokus auf die Dokumentation und die Planung gelegt. Ebenfalls haben wir uns entschieden eine Desktop Applikation zu schreiben mit JavaFX. Zusätzlich wurde ein kleiner Prototyp des Programmes erstellt. Erste Funktionen wie das Erstellen von Knoten und deren Verschiebung wurde ebenfalls realisiert.

4.3 Sprint 2

Der Sprint 2 dauerte vom 03.04 - 24.04. In diesem Sprint wurde dann richtig begonnen mit dem Programmieren der Funktionalitäten. Es wurden ebenfalls die SD/SSD Diagramme erstellt. Realisiert wurden die Funktionen zum Verbinden von Knoten und das Laden und Speichern der Knoten. Zum Laden und Speichern der Knoten haben wir uns für JAXB for Java entschieden. Dieses Framework erlaubt es aus Objekten XML Dateien zu

erstellen. In diesem Sprint haben wir bemerkt, dass unser Programm zu unübersichtlich wird, wir haben unsere Kontrollerstruktur daraufhin angepasst.

4.4 Sprint 3

Der Sprint 3 dauert vom 24.04 - 15.05. In diesem Sprint wurde vor allem zuerst ein grosses Refactoring des Codes gemacht. Der Hauptkontroller wurde in 3 Unterkontroller geteilt. Dies gab dem Programm mehr Lesbarkeit und war infolgedessen auch einfacher zu erweitern. Ebenfalls wurde das Feature zur Zusatzinformation für Knoten, der Unterknoten Button und der Typ einer Verbindung hinzugefügt. Am Schluss des Sprints wurde dann noch die Applikation optisch verschönert.

4.5 Sprint 4

Der Sprint 4 dauert vom 15.05 - 29.05. In diesem Sprint wurde das Programm beendet. Diverse Bugs wurden ausgebessert und einzelne neue Funktionen wie das Skalieren der Applikation und die ordnen Funktion wurde hinzugefügt.

4.6 Kontroller Management

Das Programm besitzt 4 Kontroller, die jeweils einer FXML Datei zugeordnet sind. Der MainController ist dabei die Schnittstelle zwischen der View und den Kontrollern. Der CanvasController ist für alle Aktionen im Programmfeld zuständig, er erstellt zum Beispiel neue Knoten oder färbt diese. Der MenuBarController ist für die Aktionen in der Menubar zuständig, er speichert und lädt die Maps. Zuletzt noch der ToolbarController, dieser prüft primär welche Aktionen im CanvasController ausgeführt werden sollen, wie z.b. Knoten erstellen, Verbinden usw.

4.7 Ordnungs Funktion

Die Knoten Ordnen Funktion wurde implementiert um dem Benutzer zu helfen sein Mindmap aufzuräumen. Die Funktion kann dabei 3 Probleme erkennen.

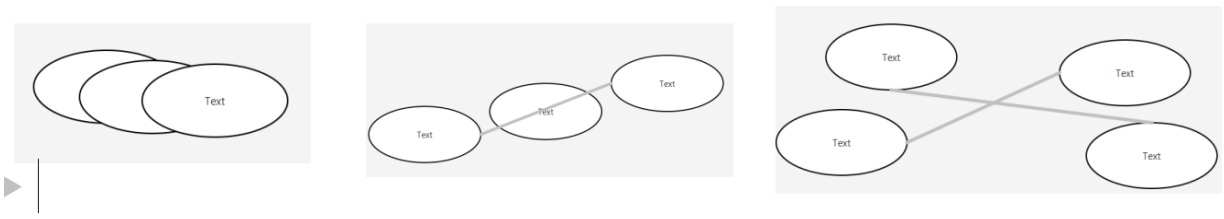

Abbildung 4.2: 3 erkennbare Probleme

Diese 3 Probleme versucht nun der Algorithmus zu lösen. Er geht dabei nach einem Try and Error konzept vor. Es wird versucht den Knoten in 8 Richtungen zu verschieben und geprüft ob das Problem gelöst wurde. Falls nicht, wird die Distanz bis auf 7x vergrössert. Die Distanz des Verschiebens ist jeweils der Radius der zu verschiebenden Ellipse.

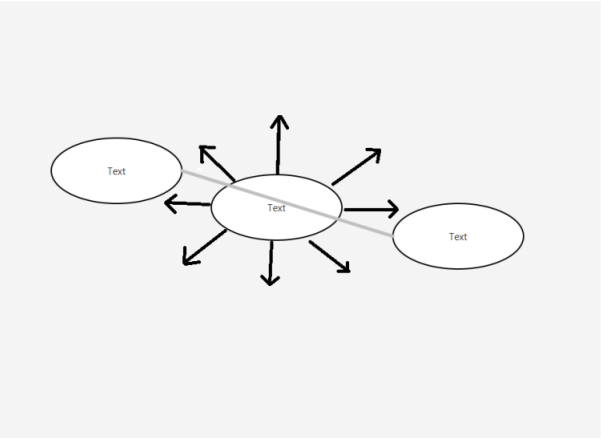

Abbildung 4.3: Problemlösung

Das Fazit zu dieser Lösung ist, dass der Algorithmus bei einfachen Problemen ohne Probleme eine Lösung findet. Bei schwereren Problemen jedoch findet er teils keine Lösung. Ebenfalls ist der Algorithmus sehr ineffizient und kann lange dauern wenn sehr viele Knoten geprüft werden müssen.

4.8 Programm ausführen

4.8.1 Tools and Software

Unsere Mindmap Applikation benutzt folgende Software und Frameworks.

- Java jdk1.8.0_191
- JavaFX http://javafx.com/javafx/8.0.191
- JavaFX Scene Builder 10.0.0

4.8.2 Installation

Um das Programm auszuführen muss zu erst das Repository per Git geklont werden.

```
git clone https://gitlab.ti.bfh.ch/rascl1/mindmap.git
```

Wenn das Projekt in ein IDE importiert wird, muss der Ordner "MindMap" als Grundlage gewählt werden. Oder das Programm kann mit der `MindMap.jar` ausgeführt werden.

5 Fazit

Das Fazit dieses Projektes ist, dass wir sehr viel gelernt haben. Es war wichtig für uns nach dem Software Engineering Modul noch einmal ein Projekt durchzuführen. Wir liefen dabei in einige Schwierigkeiten, wir hätten zum Beispiel die Kontrollerstruktur von Anfang an erstellen müssen. Es war sehr mühsam die Struktur in der Mitte der Entwicklung zu ändern. Die Applikation an sich ist nicht schlecht und wir sind relativ zufrieden mit dem Ergebnis. Das Produkt hätte aber teilweise besser implementiert werden können. Das Verbinden von Knoten hätte besser gelöst werden können, ebenfalls fühlt sich das Programm zum Teil steif an. Das Fazit ist also, dass wir sehr viel gelernt haben und somit das Projekt ein Erfolg war.

Abbildungsverzeichnis

2.1 Domainmodel . . . 3
2.2 Use Case Diagramm . . . 4
2.3 Systemkontext . . . 9

3.1 Create Node SSD . . . 11
3.2 Delete Node SSD . . . 11
3.3 Create Connection SSD . . . 12
3.4 Save Map SSD . . . 12
3.5 Load Map SSD . . . 12
3.6 Create Node SD . . . 13
3.7 Delete Node SD . . . 13
3.8 Create Connection SD . . . 14
3.9 Save Map SD . . . 14
3.10 Convert & Save Map SD . . . 15
3.11 Load Map SD . . . 16
3.12 Load & Convert Map SD . . . 17
3.13 Package Diagramm . . . 18

4.1 Use Case Diagramm . . . 19
4.2 Use Case Diagramm . . . 20
4.3 Use Case Diagramm . . . 21