Graph Visualisierung 2: vistra

Didaktische Desktop-Applikation zur Visualisierung der Traversierung von Graphen

Projekt-Dokumentation, Revision 1.8

Studiengang:	Informatik, Modul BTI-7301 Projekt 1, HS 2013/14
Autor:	Roland Bruggmann (brugr9@bfh.ch)
Betreuer:	Dr. Jürgen Eckerle, juergen.eckerle@bfh.ch
Datum:	20.12.2013

Versionen

Version	Datum	Status	Bemerkungen
1.0	04.10.2013	Inception	Requirements: Vision
1.1	11.10.2013	Inception	Project Management
1.2	18.10.2013	Inception	Requirements: Vision; Project Management
1.3	25.10.2013	Inception	Requirements, Use Cases Model: Actors, Diagram, Brief Format
1.4	01.11.2013	Inception	Requirements, Use Cases Model: Fully Dressed Format
1.5	08.11.2013	Elaboration	Design: Domain Model; Software Architecture Document
1.6	15.11.2013	Elaboration	Design, Design Model: System Sequence Diagrams, Sequence Diagrams UC1 - UC4
1.7	22.11.2013	Elaboration	Design, Design Model: Sequence Diagrams UC1 - UC6
1.8	20.12.2013	Elaboration	

Inhaltsverzeichnis

1 Requirements **1**
1.1 Vision . . . 1
1.2 Use Cases Model . . . 3
1.3 Supplementary Specification . . . 11

2 Design **13**
2.1 Domain Model . . . 13
2.2 Design Model . . . 16
2.3 Software Architecture Document . . . 29

3 Project Management **31**
3.1 Time Management . . . 31
3.2 Object Oriented Analysis and Design . . . 31
3.3 Development Environment Description . . . 31

Glossar **33**

Literaturverzeichnis **33**

Abbildungsverzeichnis **35**

Tabellenverzeichnis **37**

1 Requirements

1.1 Vision

Es soll eine Software erstellt werden, welche das Traversieren von Graphen mit verschiedenen Algorithmen darstellen kann.

Ein beliebiger Algorithmus, wie etwa derjenige von Dijkstra soll mit diesem Werkzeug auf einfache Weise visualisiert werden. Das Werkzeug soll sich als didaktisches Hilfsmittel eignen. Neue Algorithmen sollen ohne grossen Aufwand hinzugefügt werden können. Zudem soll die Software Graphen aus einer Datei importieren können.

1.1.1 Problem Statement

Daten

Das System soll folgende Graphen und Algorithmen verarbeiten können:

- Das System hält mehrere Graphen und Algorithmen als Vorgaben zur Auswahl bereit (Templates).
- Es können ungerichtete, gerichtete, ungewichtete und (positiv) gewichtete einfache Graphen (simple connected graphs, neither self-loops nor parallel edges) traversiert werden.
- Es stehen mindestens folgende, bereits implementierte Algorithmen zur Verfügung:
  - Rekursive Tiefensuche (Depth-First Search, DFS)
  - Breitensuche (Breadth-First Search, BFS)
  - Dijkstra: Suchen des kürzesten Weges zwischen zwei als Start und Ende festgelegten Knoten in einem gerichteten und gewichteten Graphen (Shortest Path)
  - Kruskal: minimaler Spannbaum berechnen (Spanning Tree)
- Nebst den als Vorgaben zur Verfügung stehenden Graphen und Algorithmen können weitere Graphen und Algorithmen importiert werden.
- Importierte Graphen und Algorithmen bleiben dem System persistent erhalten.
- Importierte Graphen und Algorithmen können auch wieder gelöscht werden.

Traversierung

- Aus einer Liste mit Graphen kann ausgewählt werden, welcher Graph traversiert werden soll.
- Ein Graph wird durch Kreise (Knoten), Geraden (ungerichtete Kanten), Pfeile (gerichtete Kanten) und Beschriftungen (Knotenbezeichnungen, Kantenbezeichnungen und Kantengewichte) dargestellt.
- Je nach Typ von Graph werden die Knoten als Kreis oder als Baum angeordnet.
- Die Anordnung der Knoten kann verändert werden: Diese können mit der Maus verschoben werden.
- Mit der Wahl des Graphen wird eine Liste mit Algorithmen erstellt und dem Benutzer zugänglich gemacht. Es sind nur diejenigen Algorithmen auswählbar, die auf den Graph-Typ angewendet werden können.
- Aus der Liste mit Algorithmen kann ausgewählt werden, welche Traversierung erfolgen soll.
- Mit der Wahl des Algorithmus wird die Berechnung der Traversierung ausgelöst.

- Die Berechnung der Traversierung erstellt eine visualisierbare Lösung.
- Mit Abschluss der Traversierung wird dem Benutzer die Visualisierung zugänglich gemacht.

Visualisierung

- Die Visualisierung kann Schrittweise erfolgen ('Step-by-step'): Dabei kann der Benutzer vor, zurück, zum Anfang oder zum Ende der Visualisierung gelangen.
- Die Schrittlänge der Visualisierung kann eingestellt werden.
- Die Visualisierung kann auch abgespielt werden ('Animation'). Dabei kann der Benutzer die Animation starten, pausieren oder stoppen.
- Das Tempo der abgespielten Visualisierung kann eingestellt werden.

1.1.2 Other Requirements and Constraints

- Zu importierende Graphen werden validiert.
- Zu importierende Algorithmen müssen ein gegebenes Interface implementieren.
- Ein Algorithmus gibt über Annotations an, welche Graph-Typen damit traversiert werden können (gerichtet, ungerichtet).
- Für ungewichtete Graphen wird ein Kanten-Gewicht der Grösse 1 angenommen, es wird also nicht unterschieden zwischen ungewichtet und gewichtet.
- Die Visualisierung zeigt Schrittweise Farbänderungen von Knoten und Kanten, evtl. auch errechnete Zwischenergebnisse.
- Mit jedem Step der Visualisierung wird auf einer Protokollpanele eine Statusmeldung ausgegeben, die die begangenen Traversierungsschritte erläutert.

1.2 Use Cases Model

1.2.1 Actors

Actor	Typ	Beschreibung
User	primary	Interagiert mit dem System.
Operating System	supporting	Dient dem System beim Zugriff auf das Dateisystem.
Graph Developer	offstage	Entwickler von Graphen in GraphML.
Algorithm Developer	offstage	Entwickler von Algorithmen in Java.

Tabelle 1.1: Actors

1.2.2 Use Cases Diagram

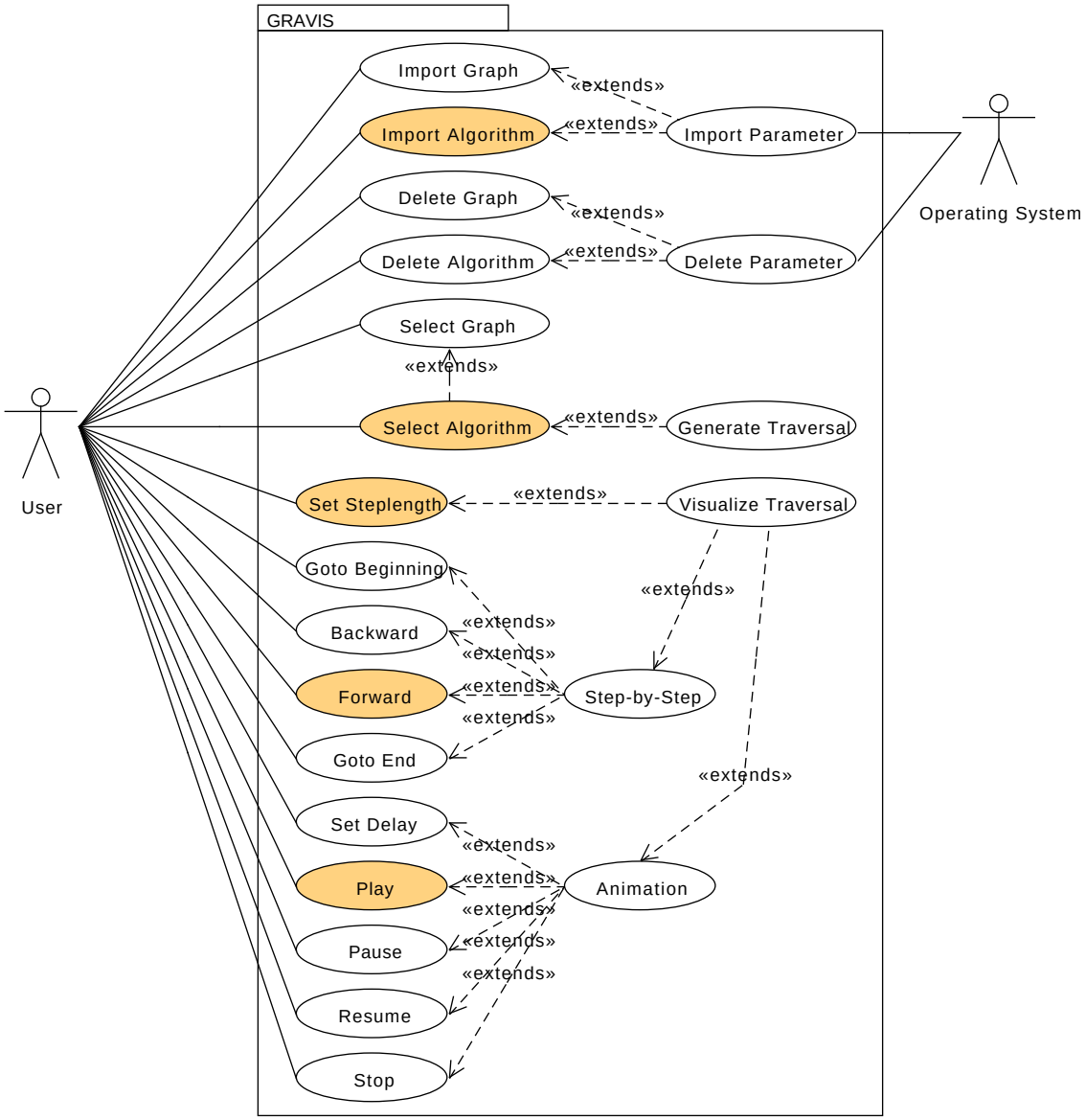

Abbildung 1.1: Use Cases Diagram

1.2.3 Use Cases in Brief Format

Import Graph: Der User kann einen neuen Graphen importieren.

(Ausgearbeitetes Format siehe Seite 5)

Import Algorithm: Der User kann einen neuen Algorithmus importieren.

(Ausgearbeitetes Format siehe Seite 6)

Delete Graph: Der User kann einen importierten Graphen löschen.

(Ausgearbeitetes Format siehe Seite 7)

Delete Algorithm: Der User kann einen importierten Algorithmus löschen.

(Ausgearbeitetes Format siehe Seite 8)

Select Graph: Der User kann einen Graphen auswählen.

(Ausgearbeitetes Format siehe Seite 9)

Select Algorithm: Der User kann einen Algorithmus auswählen und damit eine Traversierung (Traversal) generieren.

(Ausgearbeitetes Format siehe Seite 10)

Set Steplength: Der User kann für die Visualisierung die Anzahl Traversierungsschritte (Step) pro Bild (Image) einstellen.

Forward: Der User kann in der Step-by-Step-Visualisierung ein Bild vorwärts gehen.

Backward: Der User kann in der Step-by-Step-Visualisierung ein Bild rückwärts gehen.

Goto Beginning: Der User kann in der Step-by-Step-Visualisierung an das Ende springen.

Goto End: Der User kann in der Step-by-Step-Visualisierung an den Anfang springen.

Set Delay: Der User kann für die animierte Visualisierung (Animation) das Zeitintervall zwischen zwei Bildern einstellen.

Play: Der User kann die animierte Visualisierung starten.

Pause: Der User kann die animierte Visualisierung pausieren.

Resume: Der User kann die pausierte animierte Visualisierung wieder aktivieren.

Stop: Der User kann die animierte Visualisierung anhalten.

1.2.4 Use Cases in Fully Dressed Format

Die sechs UC *Import Graph*, *Import Algorithm*, *Delete Graph*, *Delete Algorithm*, *Select Graph* und *Select Algorithm* werden im ausgearbeiteten Format erläutert. Für diese UC gilt:

- Scope: System-wide
- Level: User-goal
- Primary Actor: User

UC1: Import Graph

Preconditions:

- Die Benutzerschnittstelle zur Befehlswahl ist aktiv.
- Die Dateistruktur des Betriebssystems ist zugänglich.
- Auf die zu importierende Datei sind mindestens Leserechte gesetzt.

Postconditions (success guarantee):

- Der Parameter steht dem System zur weiteren Verarbeitung zur Verfügung.
- Der Parameter steht dem User in der Parameterliste zur Auswahl bereit.
- Der Default-Graph wurde ausgewählt.

Main Success Scenario:

1. Der User wählt über die Benutzerschnittstelle das Importieren eines Graphen (import).
2. Der User wird dazu aufgefordert, den Pfad und den Dateinamen einer Datei anzugeben (open) oder den Vorgang abzubrechen (abort).
3. Die angegebene Datei wird in die Dateistruktur des Systems kopiert (copy).
4. Die angegebene Datei wird durch das System auf Kompatibilität geprüft (load).
5. Der importierte Graph wird zur Graph-Parameterliste hinzugefügt (add).
6. Der Default-Graph wird ausgewählt (select, siehe *UC5 Select Graph* Seite 9).

UC2: Import Algorithm

Preconditions:

- Die Benutzerschnittstelle zur Befehlswahl ist aktiv.
- Die Dateistruktur des Betriebssystems ist zugänglich.
- Auf die zu importierende Datei sind mindestens Leserechte gesetzt.

Postconditions (success guarantee):

- Der Parameter steht dem System zur weiteren Verarbeitung zur Verfügung.
- Der Parameter steht dem User in der Parameterliste zur Auswahl bereit.
- Der Default-Graph wurde ausgewählt.

Main Success Scenario:

1. Der User wählt über die Benutzerschnittstelle das Importieren eines Algorithm (import).
2. Der User wird dazu aufgefordert, den Pfad und den Dateinamen einer Datei anzugeben (open) oder den Vorgang abzubrechen (abort).
3. Die angegebene Datei wird in die Dateistruktur des Systems kopiert (copy).
4. Die angegebene Datei wird durch das System auf Kompatibilität geprüft (load).
5. Der importierte Algorithmus wird zur Algorithmus-Parameterliste hinzugefügt (add).
6. Der Default-Algorithm wird ausgewählt (select, siehe *UC6 Select Algorithm* Seite 10).

UC3: Delete Graph

Preconditions:

- Die Benutzerschnittstelle zur Befehlswahl ist aktiv.
- Die Dateistruktur des Betriebssystems ist zugänglich.
- Auf die zu löschende Datei sind Schreibrechte gesetzt.

Postconditions (success guarantee):

- Die Datei wurde aus dem System gelöscht.
- Die Graph-Parameterliste wurde aktualisiert.
- Der Default-Graph wurde ausgewählt.

Main Success Scenario:

1. Der User wählt ahlt über die Benutzerschnittstelle das Löschen eines Graphen (delete).
2. Der User wird dazu aufgefordert, den Pfad und den Dateinamen der Datei anzugeben (open) oder den Vorgang abzubrechen (abort).
3. Der zu löschende Graph wird aus der Graph-Parameterliste entfernt (remove).
4. Die angegebene Datei wird aus der Dateistruktur des Systems gelöscht (delete).
5. Der Default-Graph wird ausgewählt (select, siehe *UC5 Select Graph* Seite 9).

UC4: Delete Algorithm

Preconditions:

- Die Benutzerschnittstelle zur Befehlswahl ist aktiv.
- Die Dateistruktur des Betriebssystems ist zugänglich.
- Auf die zu löschende Datei sind Schreibrechte gesetzt.

Postconditions (success guarantee):

- Die Datei wurde aus dem System gelöscht.
- Die Algorithm-Parameterliste wurde aktualisiert.
- Der Default-Graph wurde ausgewählt.

Main Success Scenario:

1. Der User wählt ahlt über die Benutzerschnittstelle das Löschen eines Algorithm (delete).
2. Der User wird dazu aufgefordert, den Pfad und den Dateinamen der Datei anzugeben (open) oder den Vorgang abzubrechen (abort).
3. Der zu löschende Algorithm wird aus der Algorithm-Parameterliste entfernt (remove).
4. Die angegebene Datei wird aus der Dateistruktur des Systems gelöscht (delete).
5. Der Default-Algorithm wird ausgewählt (select, siehe *UC6 Select Algorithm* Seite 10).

UC5: Select Graph

Precondition:	Die Benutzerschnittstelle zur Wahl eines Graphen ist aktiv.

Postconditions (success guarantee):

- Der Graph wurde als ausgewählt markiert.
- Der gewählte Graph steht als geladene Instanz zur weiteren Verarbeitung zur Verfügung.
- Der gewählte Graph ist visualisiert.
- Die auf den Graphen anwendbaren Algorithmen wurden aktiv gesetzt.
- Die Parameterlisten der Benutzerschnittstellen wurden aktualisiert.

Main Success Scenario:

1. Der vormalige Graph wird im System entladen (clear).
2. Der gewählte Graph wird ins System geladen (load).
3. In der Parameterliste wird der gewählte Graph als aktuell gesetzt (set selected).
4. Die auf den Graphen anwendbaren Algorithm werden aktiv gesetzt (enable).
5. Die Parameterlisten der Benutzerschnittstellen werden aktualisiert.
6. Der aktuelle Graph wird im Visualizer dargestellt.

UC6: Select Algorithm

Preconditions:

- Die Benutzerschnittstelle zur Wahl eines Algorithm ist aktiv.

Postconditions (success guarantee):

- Der Algorithm wurde als ausgewählt markiert (*selected*).
- Der gewählte Algorithm wurde als Instanz geladen.
- Die Parameterliste der Benutzerschnittstelle wurde aktualisiert.
- Der gewählte Algorithm hat eine Traversal generiert.
- Das System ist zur Visualisierung der Traversal bereit.

Main Success Scenario:

1. In der Parameterliste wird der gewählte Algorithm als ausgewählt markiert (*selected*).
2. Der gewählte Algorithm wird ins System geladen.
3. Die Parameterliste der Benutzerschnittstelle wurde aktualisiert.
4. Der gewählte Algorithm brechnet die Traversal.
5. Der User erhält eine Mitteilung, dass die Berechnung der Traversal abgeschlossen ist.
6. Der User quittiert die Mitteilung.
7. Die Abspielkonsole zur Steuerung der Visualisierung wird aktiviert.

1.3 Supplementary Specification

Platform: Das System soll auf verschiedenen Betriebssystemen lauffähig sein, im minimum auf GNU/Linux, Mac OS und Microsoft Windows.

I18n: Das System soll in den drei Sprachen Deutsch, Französisch und Englisch bedienbar sein.

2 Design

2.1 Domain Model

2.1.1 Domain Model Diagram

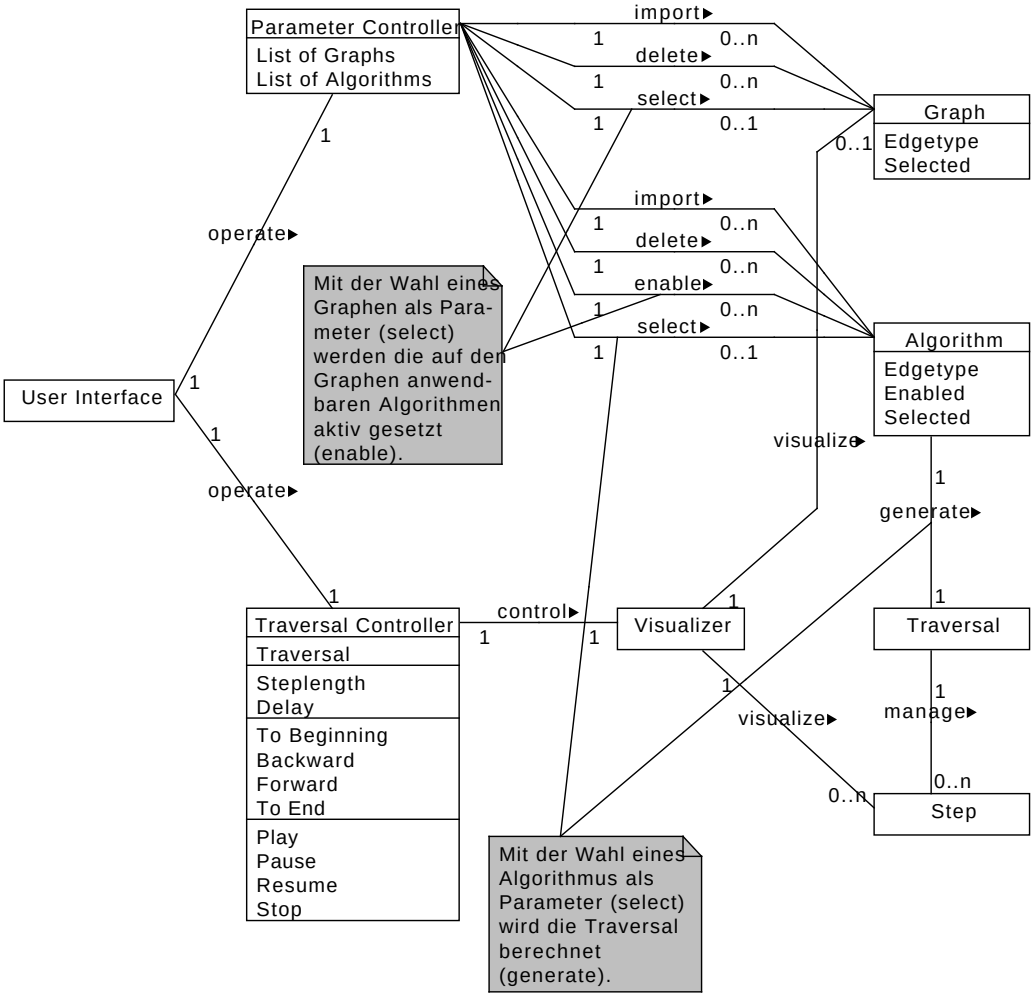

Abbildung 2.1: Domain Model Diagram

2.1.2 Domain Model Description

Es folgt eine Beschreibung der Konzeptklassen (conceptual classes) mit Assoziationen, wie im Domain Model Diagram gezeigt. Multiplizitäten und Attribute werden in Klammern angegeben.

User Interface

- Über ein User Interface (1) kann ein User einen Parameter Controller (1) bedienen (*operate*).
- Über ein User Interface (1) kann ein User einen Traversal Controller (1) bedienen (*operate*).

Parameter Controller

- Ein Parameter Controller verwaltet die zur Traversal benötigten Parameter Graph (*List of Graphs*) und Algorithm (*List of Algorithms*).
- Der Parameter Controller hält eine gegebene Anzahl von Graphen als Vorlagen bereit.
- Per Parameter Controller (1) kann ein User einen oder mehrere Graphen (0..n) des Formates *.graphml importieren (*import*). Die importierten Graphen werden der Liste mit Graphen (*List of Graphs*) hinzugefügt.
- Per Parameter Controller (1) kann ein User vormals importierte Graphn (0..n) wieder löschen (*delete*). Diese werden aus der Liste mit Graphen (*List of Graphs*) wieder entfernt.
- Per Parameter Controller (1) kann ein User einen Graphen (0..1) als Parameter auswählen (*select*).
- Mit der Wahl eines Graphen (1) als Parameter werden die auf den Graphen anwendbaren Algorithmen (0..n) aktiv gesetzt (*enable*).
- Ein Parameter Controller hält eine gegebene Anzahl Algorithmen als Vorlagen bereit.
- Per Parameter Controller (1) kann ein User einen oder mehrere Algorithmen (0..n) importieren (*import*), sofern diese ein gefordertes Interface implementieren. Die importierten Algorithmen werden der Liste mit Algorithmen (*List of Algorithms*) hinzugefügt.
- Per Parameter Controller (1) kann ein User vormals importierte Algorithmen (0..n) wieder löschen (*delete*). Diese werden aus der Liste mit Algorithmen (*List of Algorithms*) wieder entfernt.
- Per Parameter Controller (1) kann ein User einen aktivierten (*Enabled*) Algorithmus (0..1) als Parameter auswählen (*select*).

Graph

- Ein Graph hat ungerichtete oder gerichtete Kanten (*Edgetype*).
- Ein Graph kann ausgewählt werden (*Selected*).

Algorithm

- Ein Algorithm kann Graphen mit ungerichteten oder gerichteten Kanten (*Edgetype*) verarbeiten.
- Ein Algorithm (1) generiert eine Traversal (1) (*generate*).
- Ein Algorithm kann aktiviert werden (*Enabled*).
- Ein aktivierter Algorithm (*Enabled*) kann ausgewählt werden (*Selected*).
- Infolge des Auswählens eines Algorithm (1) als Parameter (*Selected*) wird eine Traversal (1) erstellt (*generate*).

Traversal

- Eine Traversal (1) kann einen oder mehrere Steps (0..n) verwalten (*manage*).

Step

- Ein Step ist ein Schritt der Traversierung.

Traversal Controller

- Ein Traversal Controller (1) steuert einen Visualizer (1) (*control*).
- Der Traversal Controller hält eine Traversierung (*Traversal*).
- Per Traversal Controller kann ein User die Anzahl Schritte pro Bild (*Steplength*) einstellen.
- Per Traversal Controller kann ein User die Zeit zwischen zwei Bildern (*Delay*) einstellen.
- Per Traversal Controller kann ein User Step-by-Step-Elemente (*Forward, Backward, To Beginning, To End*) bedienen.
- Per Traversal Controller kann ein User Animations-Elemente (*Play, Pause, Resume, Stop*) bedienen.

Visualizer

- Ein Visualizer (1) kann einen Graphen (0..1) visualisieren (*visualize*).
- Ein Visualizer (1) kann einen oder mehrere Steps (0..n) visualisieren (*visualize*).

2.2 Design Model

2.2.1 Systemkomponenten

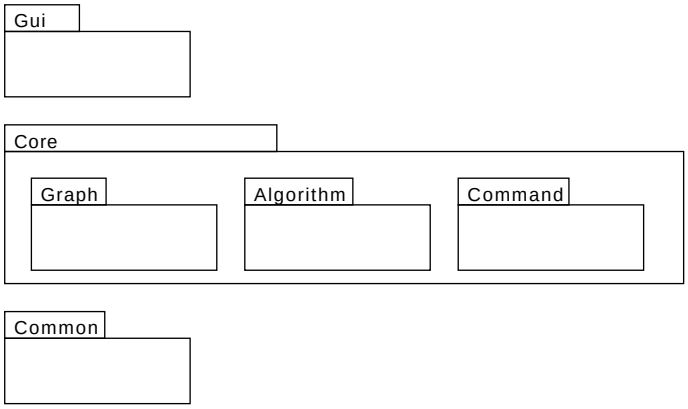

Abbildung 2.2: Software Architecture, Package Diagram

Die Systemkomponenten sind in Schichten unterteilt, wobei nur eine höher liegende Schicht direkten Zugriff auf eine darunterliegende Schicht hat (Schichtenarchitektur). Für die Architektur lassen sich von (unten nach oben) grob die Systemkomponenten *Common*, *Core* und *Gui* identifizieren. Die Drei-Schichten-Architektur ergibt sich aus den Requirements:

- Die Komponente Common hält die Interfaces für einen Actor 'Algorithm Developer' (offstage) bereit,
- die Komponente Gui enthält die (grafische) Benutzerschnittstelle und
- die Komponente Core kümmert sich um das Initialisieren sämtlicher Komponenten, I/O, Verwaltung der Parameter sowie der Berechnung der Traversierung.

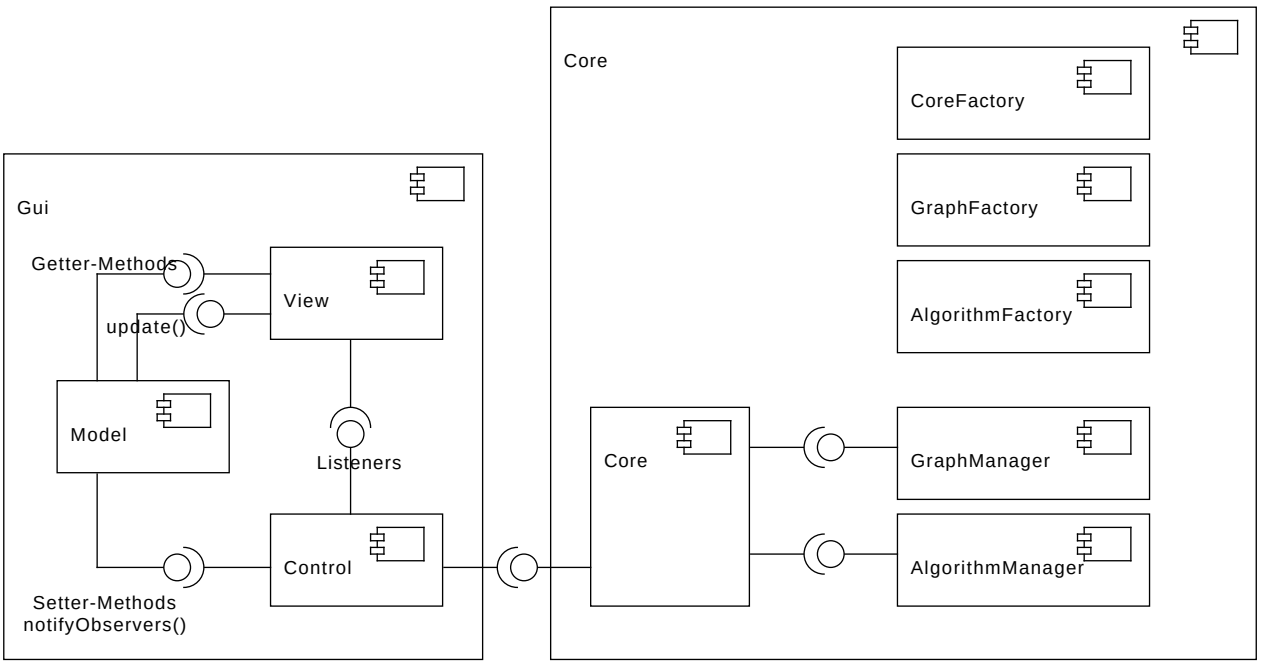

Abbildung 2.3: Software Architecture, Component Diagram

Common

Die Komponente hält die für die Implementation eines Algorithmus zu verwendende Schnittstellen bereit. Diese sind für Algorithmen zu verwenden, welche importiert werden wollen.

Core

Die Komponente implementiert:

- Data Model: Datenhaltung für Graph (Datenelemente Knoten und Kanten), Algorithmus und berechnete Traversierung, Traversierungsschritte als Resultat der Operation eines Algorithmus auf einen Graphen
- Business Logik:
  - Handling von Daten-Import und Löschen von Daten
  - Validierung Graphen und Algorithmen beim Import
  - Handling von Graphen und Algorithmen
  - Traversierung und damit Erstellen der visualisierbaren Lösung
- Core Inteface: eine Schnittstelle, welche der Komponente GUI zur Verfügung steht

Gui

Die Komponente implementiert ein Model-View-Control (MVC) unter Verwendung des Java-Observer-Pattern:

- Model: Observable mit Attributen und deren getter- und setter-Methoden
- View: Observer mit grafischen Elementen wie z.B. Menubar, Knöpfe, Regler und Text-Panelen
- Control: Implementiert Listeners mit deren Methoden und evtl. private Methoden

2.2.2 Elaboration

Zur Elaboration des Designs werden die sechs Use Cases *Import Graph*, *Import Algorithm*, *Delete Graph*, *Delete Algorithm*, *Select Graph* und *Select Algorithm* erarbeitet und je mit einem System Sequence Diagram (SSD), einem Sequence Diagram (SD) und einem Design Class Diagram (DCD) illustriert.

- Ein System Sequence Diagram zeigt, wie eine Benutzerinteraktion vom System verarbeitet wird.
- Ein Sequence Diagram zeigt, wie Objekte miteinander arbeiten und erläutert die Zuständigkeiten der Klassen.
- Ein Design Class Diagram zeigt die Elemente des Systems unter Berücksichtigung ihrer Funktionalität.

Für die Use Cases *Select Graph* und *Select Algorithm* wird zudem die Konzeptklasse *Traversal Controller* als State Machine (SM) betrachtet und mit einem State Diagram illustriert.

Ein weiteres State Diagram zeigt die Konzeptklasse *Parameter Controller* als State Machine. Dieses illustriert die Use Cases *Set Steplength*, *Set Delay*, *Forward*, *Backward*, *Goto Beginning*, *Goto End*, *Play*, *Pause*, *Resume* und *Stop*.

Generell implementiert ein Controller ausschliesslich System-Operationen (und eventuell einige private Methoden). Jeder System-Event wird mit einer einzigen System-Operation eines Controllers assoziiert. Controller sind im System vistra die implementierten Klassen *gui::Control* und *core::Core*.

UC1 Import Graph

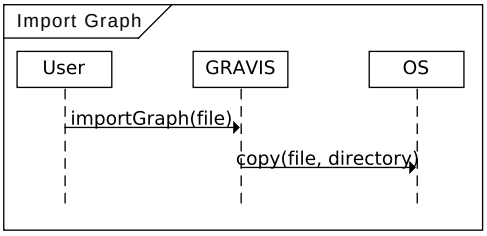

Abbildung 2.4: UC1 Import Graph, System Sequence Diagram

UC2 Import Algorithm

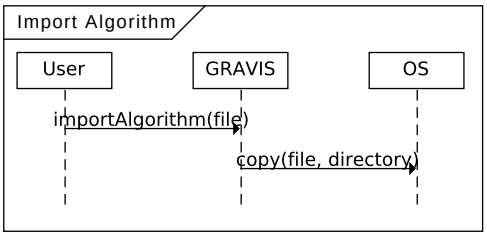

Abbildung 2.7: UC2 Import Algorithm, System Sequence Diagram

UC3 Delete Graph

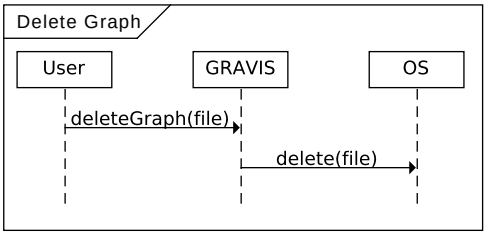

Abbildung 2.10: UC3 Delete Graph, System Sequence Diagram

UC4 Delete Algorithm

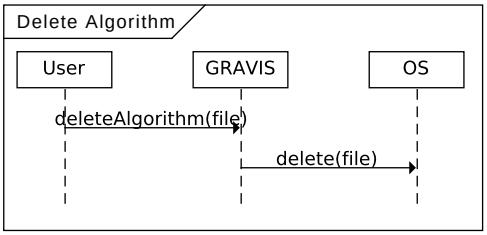

Abbildung 2.13: UC4 Delete Algorithm, System Sequence Diagram

Parameter Controller

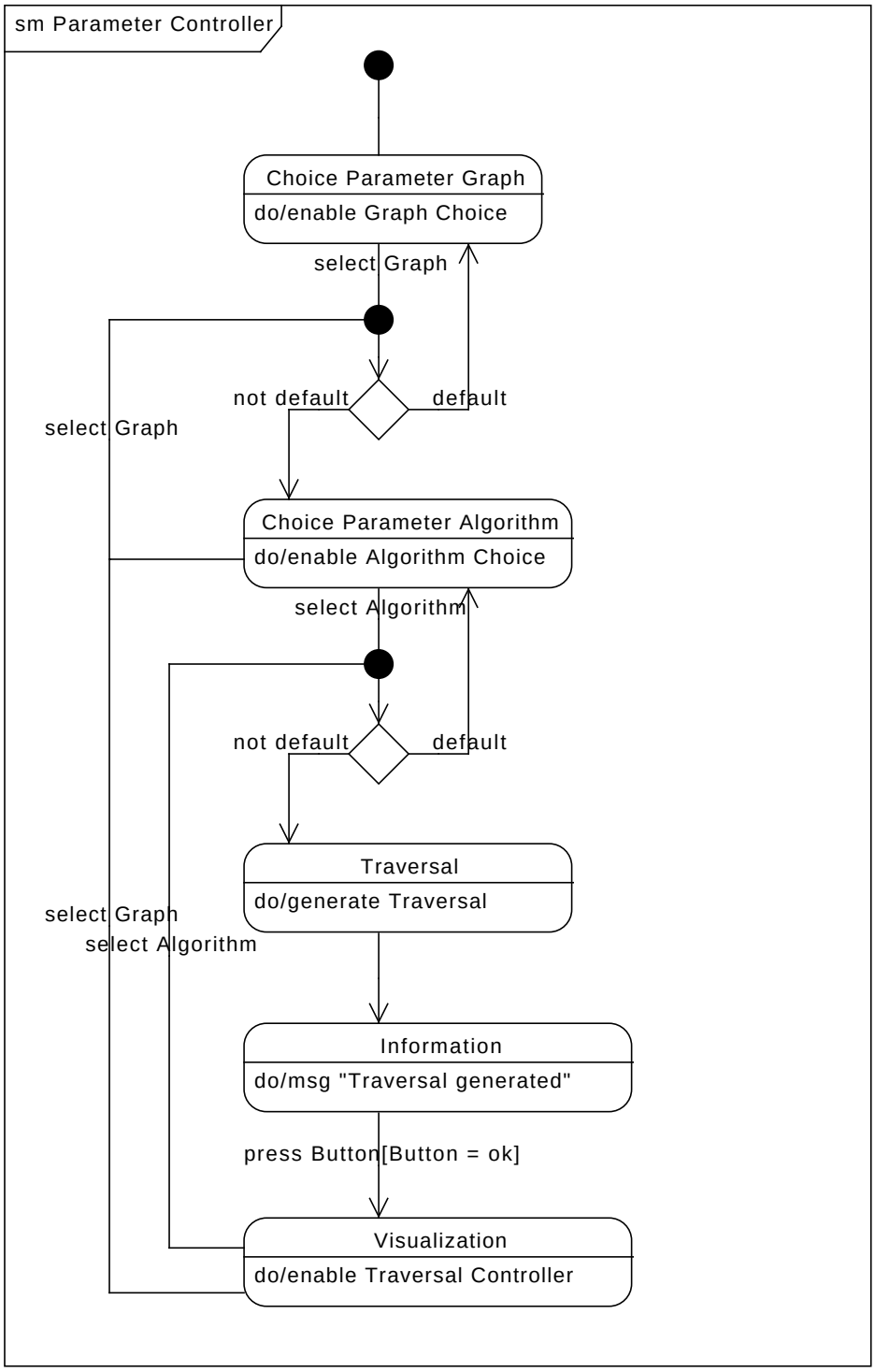

Abbildung 2.16: Parameter Controller, State Diagram

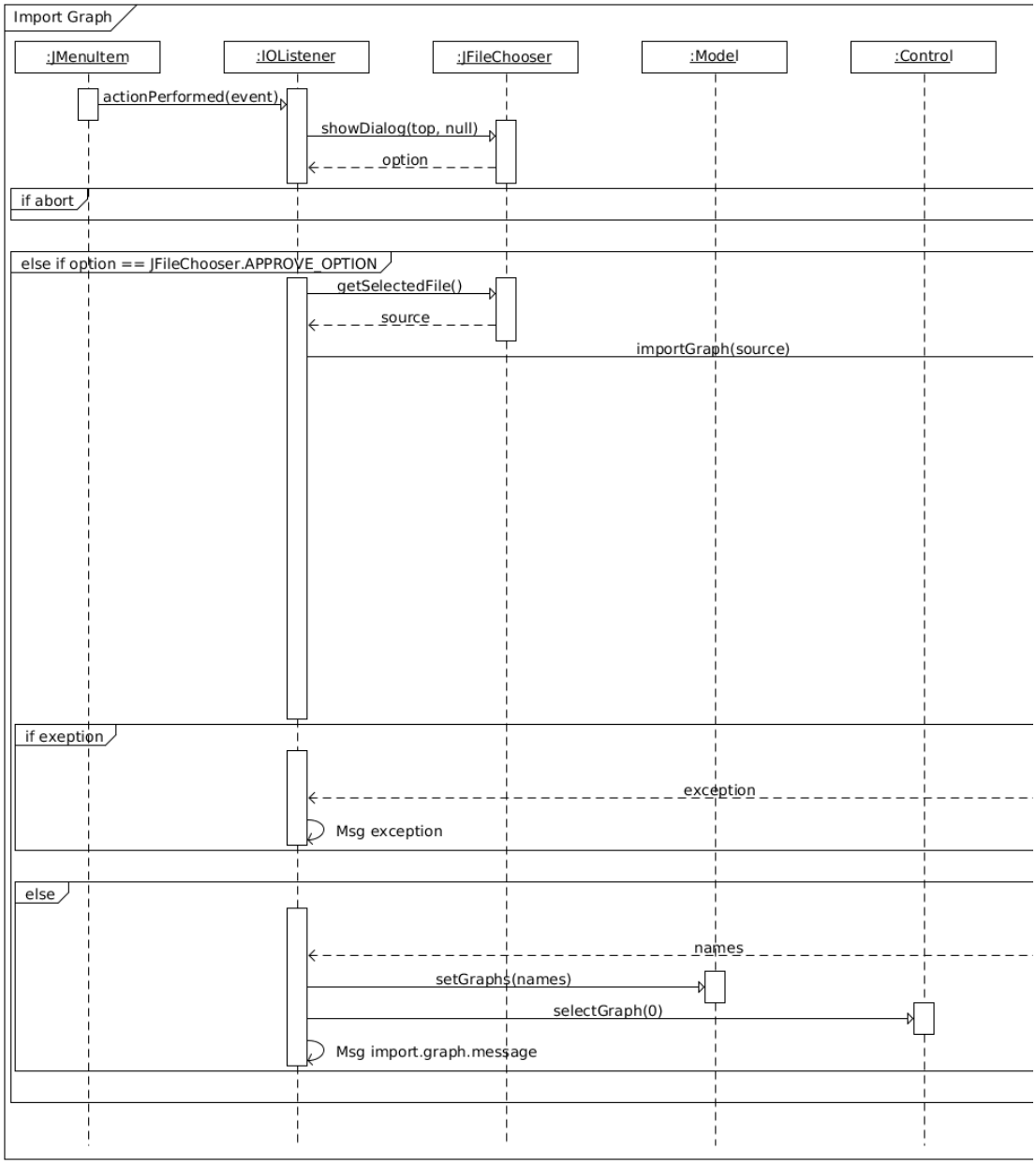

Abbildung 2.5: UC1 Import Graph, Sequence Diagram 1/2

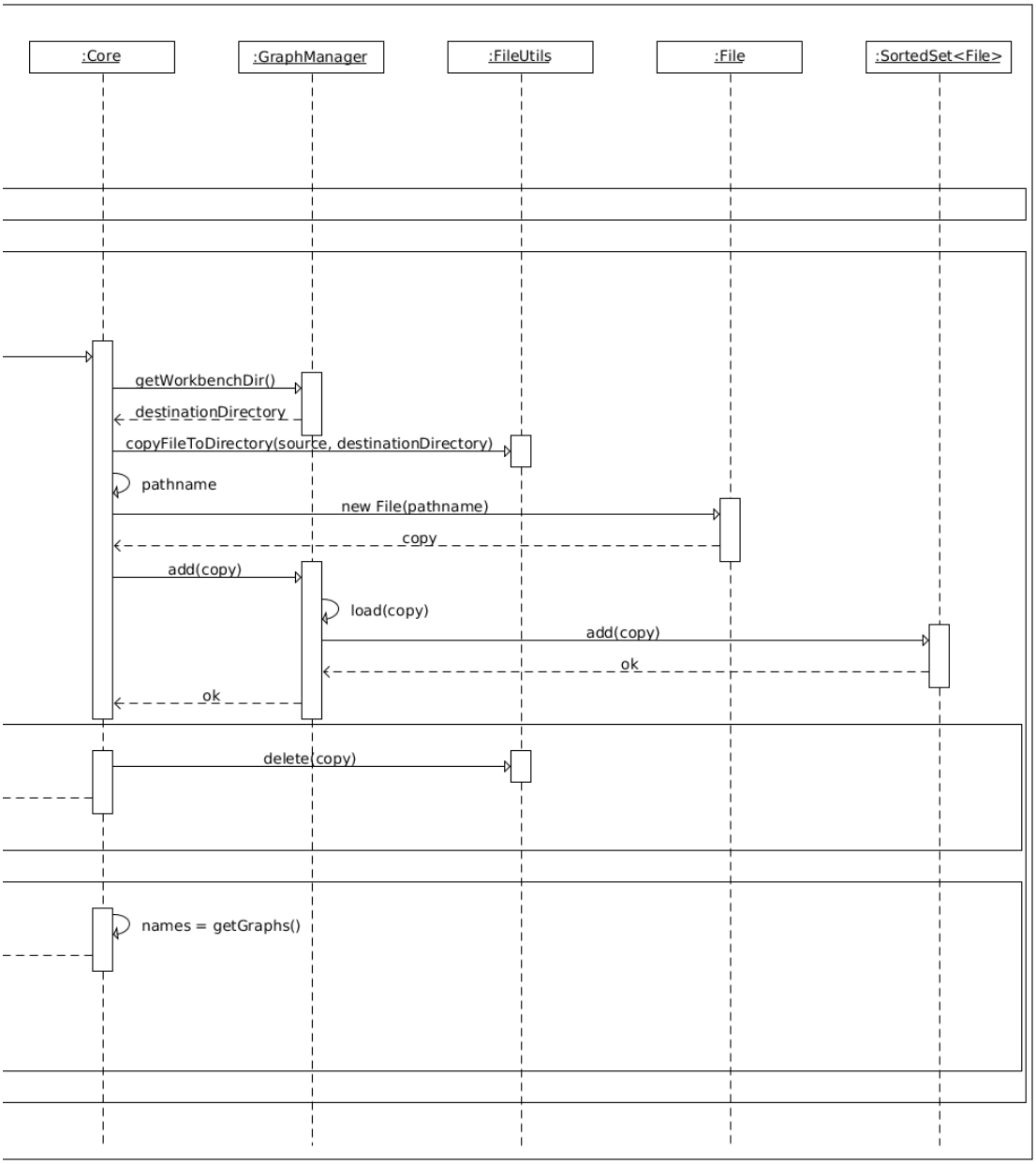

Abbildung 2.6: UC1 Import Graph, Sequence Diagram 2/2

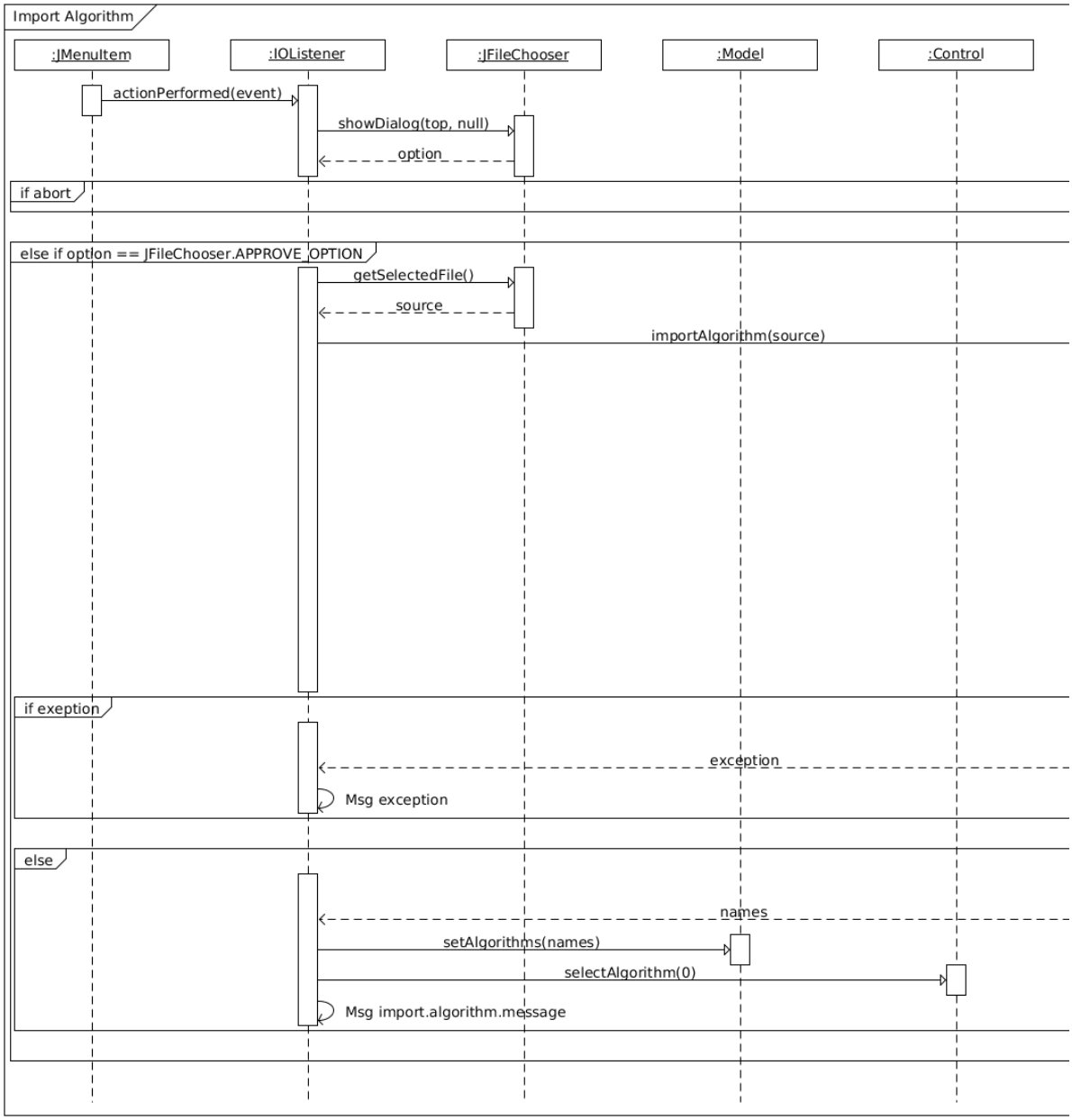

Abbildung 2.8: UC2 Import Algorithm, Sequence Diagram 1/2

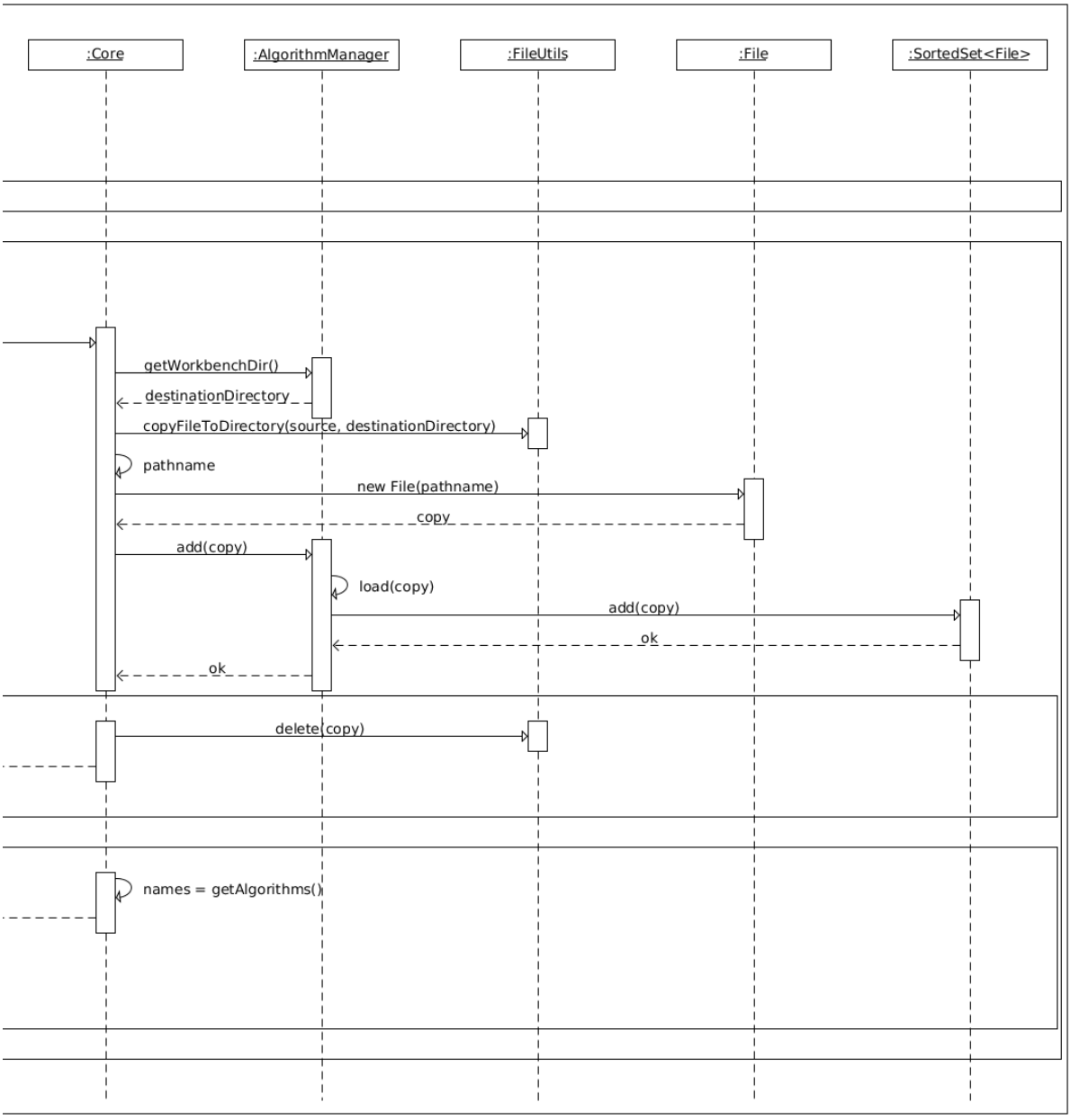

Abbildung 2.9: UC2 Import Algorithm, Sequence Diagram 2/2

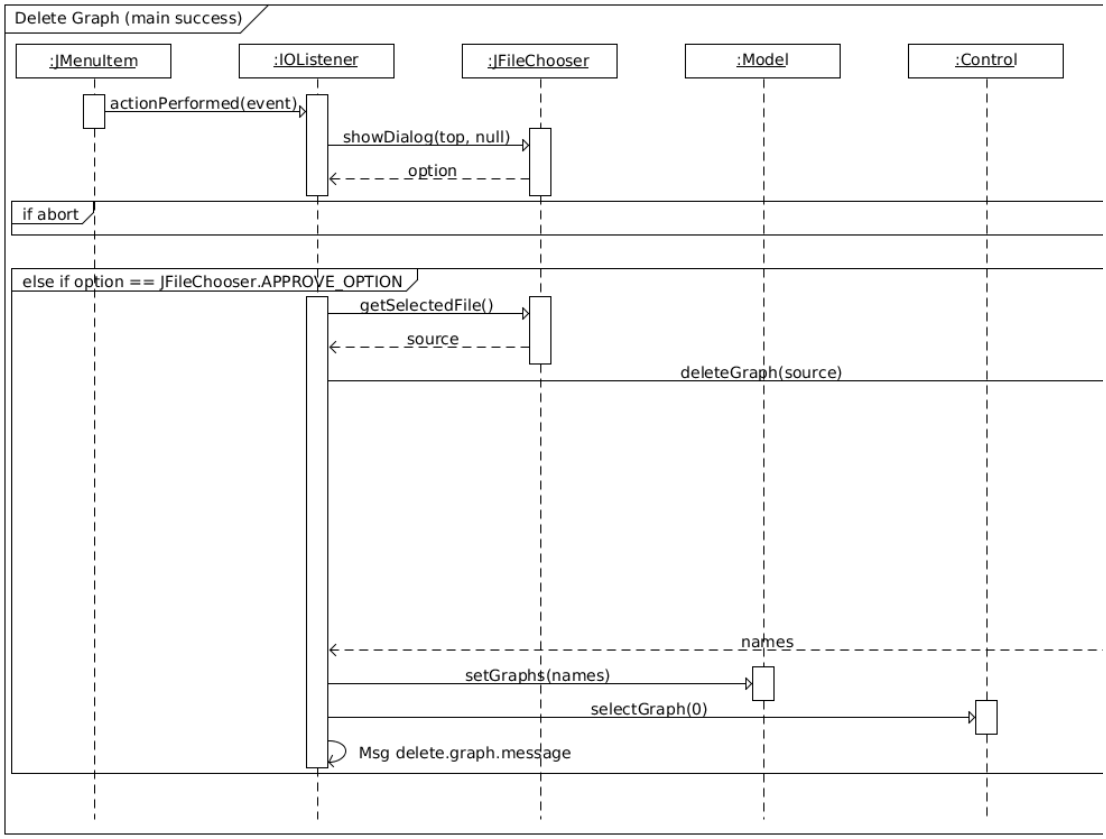

Abbildung 2.11: UC3 Delete Graph, Sequence Diagram 1/2

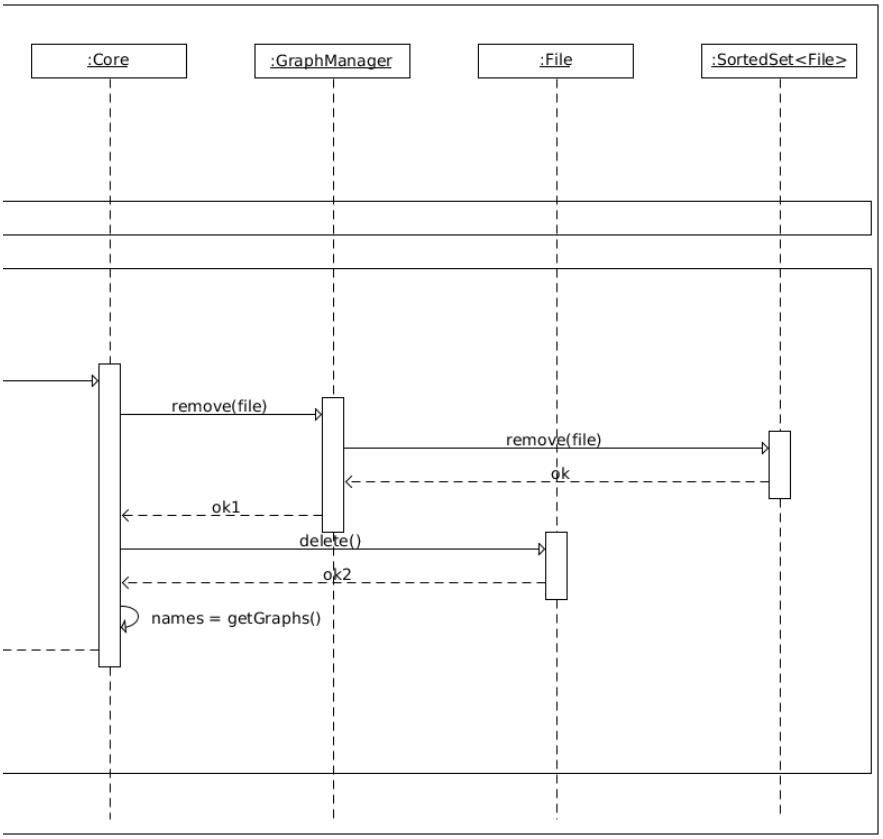

Abbildung 2.12: UC3 Delete Graph, Sequence Diagram 2/2

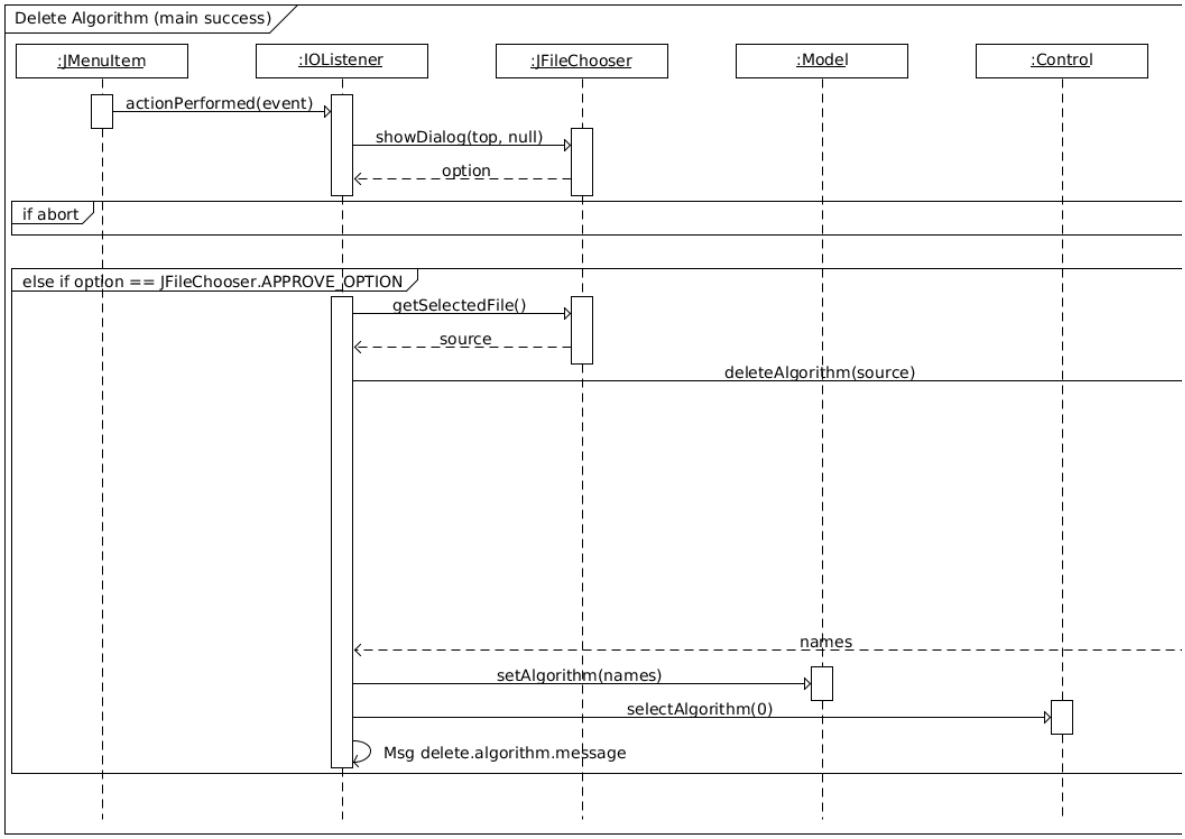

Abbildung 2.14: UC4 Delete Algorithm, Sequence Diagram 1/2

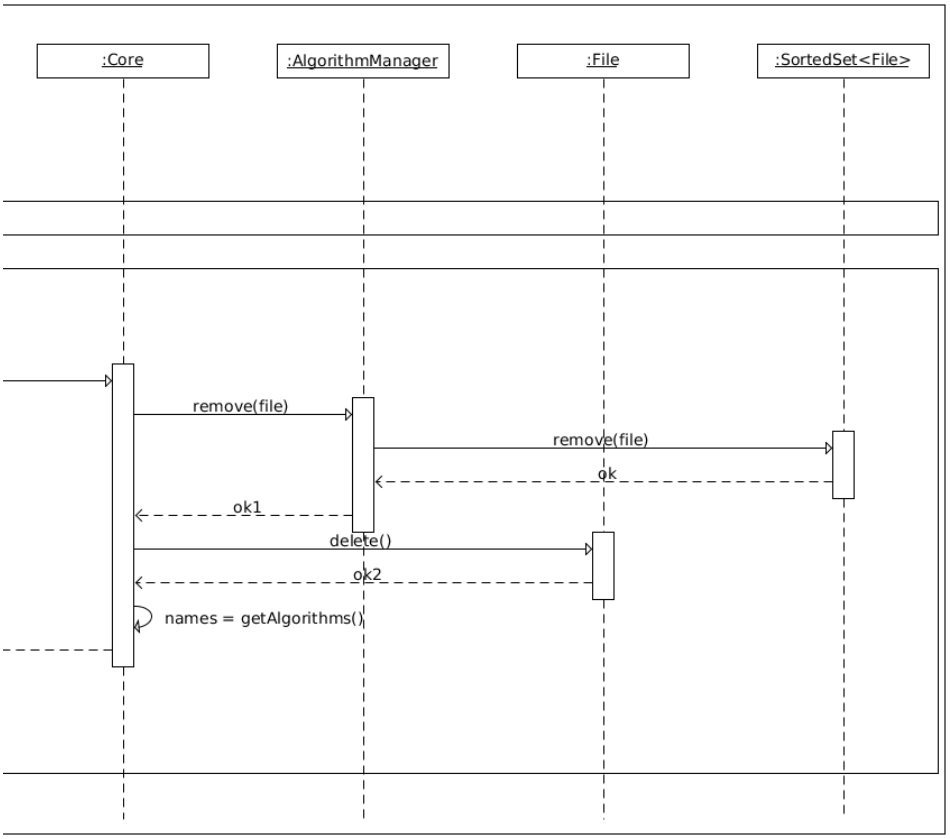

Abbildung 2.15: UC4 Delete Algorithm, Sequence Diagram 2/2

UC5 Select Graph

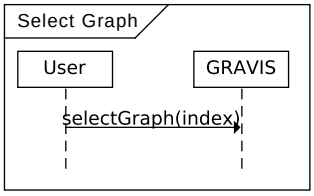

Abbildung 2.17: UC5 Select Graph, System Sequence Diagram

UC6 Select Algorithm

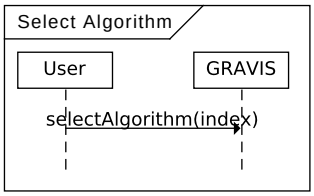

Abbildung 2.18: UC6 Select Algorithm, System Sequence Diagram

Traversal Controller

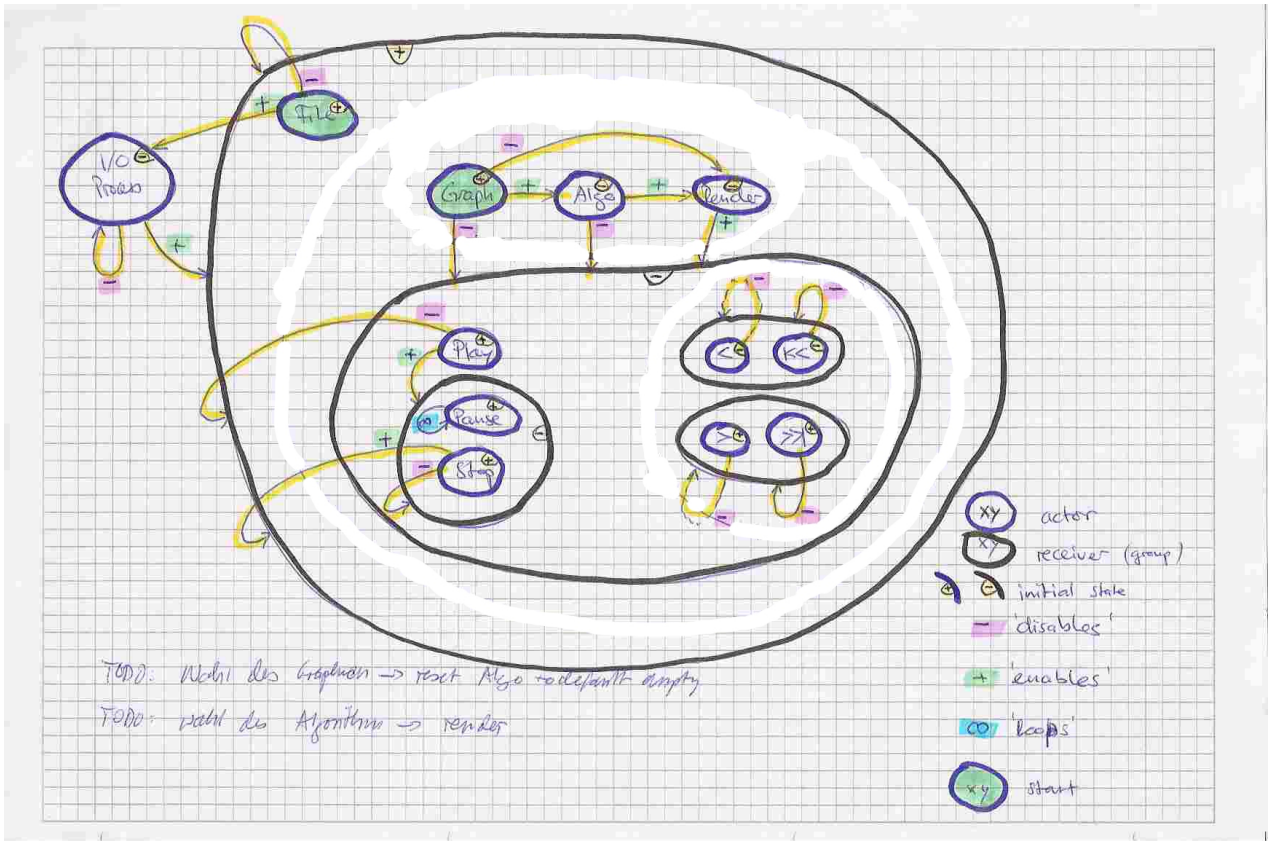

Abbildung 2.19: Traversal Controller, State Diagram

2.3 Software Architecture Document

2.3.1 Architektonische Entscheidungen

Matrixmultiplikation

Brainstorming zu Beginn des Projektes: Kybernetisches System mit

- Operand: Graph als Adjazenz-Matrix
- Operator: e.g. Dijkstra-Algorithmus (als Matrix)
- Transition: Ein Schritt der Traversierung
- Transformierte: Graph im neuen Zustand

Berechnung der Traversierung per Matrixmultiplikation:

- Definition der Operatoren (Algorithmen) allgemein gültig als Matrizen
- Anwenden der Operatoren (Algorithmen) auf Operanden (Graphen) als Matrixmultiplikation
- Resultat: Liste von (bijektiven, also eineindeutigen) Transformationen ('Steps') als Matrizenmechanik

Entscheidungen

- Themenkreis Matrizen: Entscheid, dass keine Matrizen-Multiplikation. Implementation mit Bezug auf *Algorithm Design* von M. Goodrich und R. Tamassia [4].
- Zu Projektbeginn wurde relativ bald eine Drei-Schichten-Architektur entworfen.
- In der Woche 45 wurde entschieden, das Projekt als Maven-Projekt zu halten. Der Entscheid wurde vorallem getroffen, da Properties zentral verwaltet und Tests sauber vom restlichen Sourcecode getrennt gehalten werden können.

2.3.2 Process View

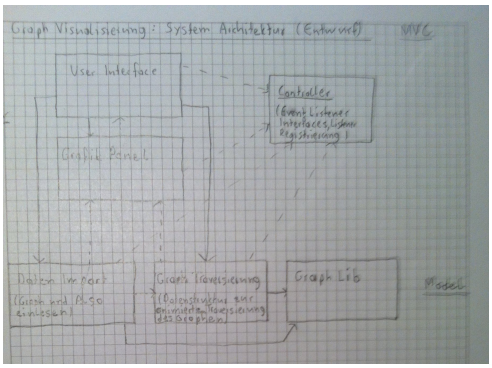

Abbildung 2.20: Gui, MVC: Conceptual Class Diagram

2.3.3 Use-Case View

Folgende Use Cases wurden implementiert:

- Daten:
  - Neuen Graphen oder Algorithmus importieren
  - Importierter Graph oder Algorithmus löschen

- Traversierung:
  - Graph auswählen
  - Algorithmus auswählen
- Visualisierung:
  - Einstellen Steplength: Anzahl Traversierungs-Schritte pro Bild
  - Einstellen Delay: Zeitintervall zwischen zwei Bildern (in Sekunden)
  - Visualisierung, Step-by-Step: Ein Bild vor, ein Bild zurück, an das Ende oder den an den Anfang springen
  - Visualisierung, Animation: Starten, Anhalten, Stoppen

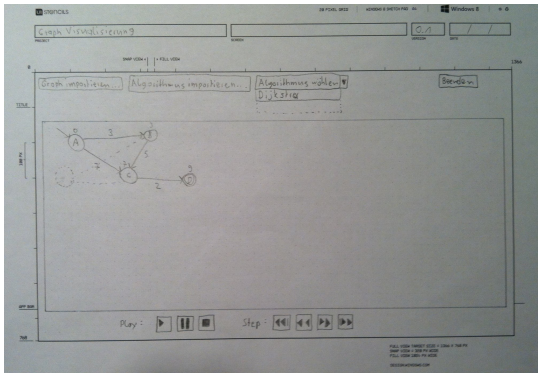

Abbildung 2.21: Gui, View: Screen Sketch

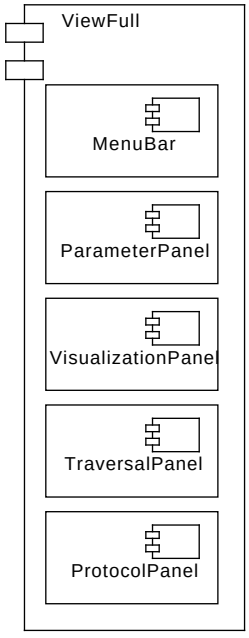

Abbildung 2.22: Gui, View: Component Diagram

3 Project Management

3.1 Time Management

Das Modul BTI-7301 Projekt 1 startete mit Beginn des Herbstsemester 2013/14. In der Woche 38 wurden die Projekte durch die Dozenten vorgestellt. Es wurden Teams gebildet und den den Projekten bzw. den Dozenten zugeteilt. Ein erstes Treffen des zuständigen Dozenten mit dem Team fand statt und erste Vereinbarungen wurden getroffen. Der Zeitplan gliedert sich nun wie folgt in vier Phasen:

Phase 1: Projektplanung und Systemarchitektur Wochen 39/40/41(/42) 2013 (3-4 Wochen)

- Einarbeiten in die Thematik (Graphen, Algorithmen, ...)
- Erstellen der Requirements, Spezifikation
- Design der Systemarchitektur

Phase 2: Wöchentliche Sprintzyklen Wochen (42/)43 bis 51 2013 (9-10 Wochen)

- Use Case wählen für nächsten Sprint
- (Re-)Design der Interfaces und Klassenhierarchie, Test und Implementation
- Dokumentation

Phase 3: Projektabschluss Wochen 52 2013 und 01 2014 (2 Wochen)

- Dokumentation abschliessen
- Erstellen der Präsentation

Phase 4: Präsentation des Projektes Wochen 02 und 03 2014 (2 Wochen)

- Besprechen des Ablaufes der Präsentation
- Checklisten Medien und Geräte
- Präsentation

3.2 Object Oriented Analysis and Design

Die Entwicklung des Projektes erfolgt objektorientiert und wird laufend dokumentiert. Die Elaboration der Komponenten und der Aufbau der Dokumentation richten sich nach C. Larman [3]. Für die Formulierung der Use Cases im ausgearbeiteten Format wurde ein LaTeX-Style-File [1] erstellt und verwendet.

3.3 Development Environment Description

3.3.1 Programming Language and Libraries

Das System wird in der Programmiersprache Java implementiert. Es werden das Java Universal Network/Graph Framework JUNG [6], das XML-basierte Format GraphML [2], die Generic Method Pattern von Roberto Tamassia, Michael Goodrich und Eric Zamor [5], die Apache Commons IO Library und das Java Swing Framework verwendet. Als integrierte Entwicklungsumgebung kommt die Software Eclipse IDE zum Einsatz.

3.3.2 Sourcecode Management

Für das Sourcecode Management (SCM) wird die Software git verwendet, das Projekt wird auf der webbasierten Plattform *github* gehalten. In der Woche 45 wurde entschieden, das Projekt als Maven-Projekt zu halten. In der Woche 51 wurde entschieden, dass die Teammitglieder je ein Projekt alleine finalisieren.

- Ab Woche 39: `https://github.com/brugr9/ch.bfh.bti7301.hs2013.GraphVisualisierung2`
- Ab Woche 45: `https://github.com/brugr9/gravis`
- Ab Woche 51: `https://github.com/brugr9/vistra`

Die Vorgänger-Projekte sind weiterhin bis mindestens nach Erhalt der Modul-Bewertung erreichbar.

Literaturverzeichnis

[1] R. E. Bruggmann, "ooaduc.sty - LaTeX-Style File for Use Cases in Object-Oriented Analysis and Design." [Online]. Available: https://gist.github.com/brugr9/5178487

[2] GraphML-Team, "The GraphML File Format." [Online]. Available: http://graphml.graphdrawing.org/

[3] C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*, 3rd ed. Prentice Hall, 2004.

[4] R. T. Michael Goodrich, *Algorithm Design*. John Wiley and Sons, Inc., 2002.

[5] E. Z. Roberto Tamassia, Michael Goodrich, "net.datastructures." [Online]. Available: http://net3.datastructures.net/

[6] The-JUNG-Framework-Development-Team, "Java Universal Network/Graph Framework JUNG." [Online]. Available: http://jung.sourceforge.net/

Abbildungsverzeichnis

1.1 Use Cases Diagram . . . 3

2.1 Domain Model Diagram . . . 13
2.2 Software Architecture, Package Diagram . . . 16
2.3 Software Architecture, Component Diagram . . . 16
2.4 UC1 Import Graph, System Sequence Diagram . . . 18
2.7 UC2 Import Algorithm, System Sequence Diagram . . . 18
2.10 UC3 Delete Graph, System Sequence Diagram . . . 18
2.13 UC4 Delete Algorithm, System Sequence Diagram . . . 18
2.16 Parameter Controller, State Diagram . . . 19
2.5 UC1 Import Graph, Sequence Diagram 1/2 . . . 20
2.6 UC1 Import Graph, Sequence Diagram 2/2 . . . 21
2.8 UC2 Import Algorithm, Sequence Diagram 1/2 . . . 22
2.9 UC2 Import Algorithm, Sequence Diagram 2/2 . . . 23
2.11 UC3 Delete Graph, Sequence Diagram 1/2 . . . 24
2.12 UC3 Delete Graph, Sequence Diagram 2/2 . . . 25
2.14 UC4 Delete Algorithm, Sequence Diagram 1/2 . . . 26
2.15 UC4 Delete Algorithm, Sequence Diagram 2/2 . . . 27
2.17 UC5 Select Graph, System Sequence Diagram . . . 28
2.18 UC6 Select Algorithm, System Sequence Diagram . . . 28
2.19 Traversal Controller, State Diagram . . . 28
2.20 Gui, MVC: Conceptual Class Diagram . . . 29
2.21 Gui, View: Screen Sketch . . . 30
2.22 Gui, View: Component Diagram . . . 30

Tabellenverzeichnis

1.1 Actors 3
UC1: Import Graph 5
UC2: Import Algorithm 6
UC3: Delete Graph 7
UC4: Delete Algorithm 8
UC5: Select Graph 9
UC6: Select Algorithm 10