Berner Fachhochschule
Technik und Informatik

Projekt 1

Modul BTI-7301, Herbstsemester 2013/14

# Graph Visualisierung 2: GRAVIS

Projekt-Dokumentation, Revision 1.5

Studenten: Roland Bruggmann, brugr9@bfh.ch
Patrick Kofmel, kofmp1@bfh.ch

Dozent: Dr. Jürgen Eckerle, juergen.eckerle@bfh.ch

Datum: November 11, 2013

Didaktische Desktop-Applikation zur Visualisierug von Graphen und deren Traversierung unter Anwendung von Graphen-Algorithmen.

# Contents

# 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 so 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

Es stehen verschiedene Graphen und Algorithmen als Vorgaben zur Verfügung.

- Es können gerichtete, ungerichtete, gewichtete und ungewichtete Graphen mit Einfach- und Mehrfachkanten traversiert werden.
- Nebst den im System als Vorgaben zur Verfügung stehenden Graphen können weitere Graphen importiert werden.
- Es stehen mindestens folgende, bereits implementierte Algorithmen zur Verfügung:
  - Dijkstra: Suchen des kürzesten Weges zwischen zwei als Start und Ende festgelegten Knoten in einem gerichteten und gewichteten Graphen
  - Kruskal: minimaler Spannbaum berechnen
  - Rekursive Tiefensuche
  - Breitensuche
- Nebst den im System als Vorgaben zur Verfügung stehenden Algorithmen können weitere Algorithmen importiert werden.
- Importierte Daten bleiben dem System persistent erhalten.
- Importierte Daten 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 zuggänglich 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.
- Für manche Algorithmen muss ein Start-, z.T. auch ein Endknoten angegebenn werden.
- Mit der Wahl des Algorithmus (und evtl. von Start- resp. Endknoten) wird die Traversierung ausgelöst.
- Die 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. 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, gewichtet, ungewichtet).
- 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. |
| Algorithm Developer | offstage | Entwickler von Algorithmen. |

Table 1.1.: Actors

### 1.2.2. Use Cases Diagram

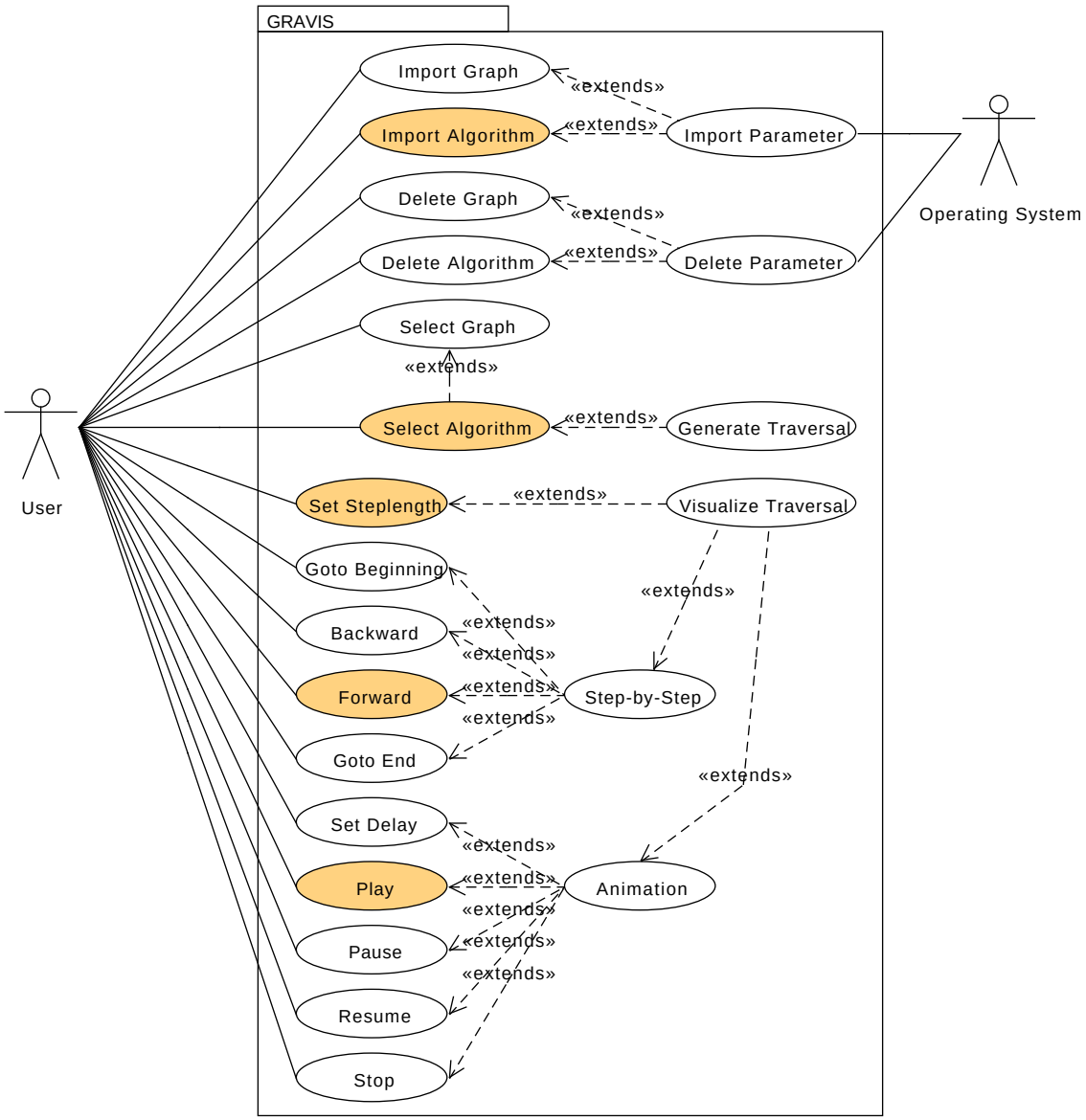


Figure 1.1.: Use Cases Diagram

### 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.

(Ausgearbeitetes Format siehe Seite 10)

**Set Steplength:** Der User kann für die Visualisierung die Anzahl Traversierungsschritte pro Bild 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 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.
- Die Parameterlisten wurden auf Default-Werte gesetzt.
- Die Parameterlisten der Benutzerschnittstellen wurden aktualisiert.
- Der Graph wurde durch das System ausgewählt und im GUI dargestellt.

*Main Success Scenario:*

1. Der User startet den Graph-Import.
2. Der User wird dazu aufgefordert, den Pfad und den Dateinamen einer Datei anzugeben oder den Vorgang abzubrechen.
3. Die angegebene Datei wird in die Dateistruktur des Systems kopiert.
4. Die angegebene Datei wird durch das System auf Kompatibilität geprüft.
5. Der zu importierte Graph wird zur Graph-Parameterliste hinzugefügt.
6. Der Graph wird durch das System in der Graph-Parameterliste ausgewählt (siehe *UC 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.
- Die Graph-Parameterliste wurde aktualisiert.
- Die Parameterlisten wurden auf Default-Werte gesetzt.
- Die Parameterlisten der Benutzerschnittstellen wurden aktualisiert.

*Main Success Scenario:*

1. Der User startet den Algorithmus-Import.
2. Der User wird dazu aufgefordert, den Pfad und den Dateinamen einer Datei anzugeben oder den Vorgang abzubrechen.
3. Die angegebene Datei wird in die Dateistruktur des Systems kopiert.
4. Die Datei wird durch das System auf Kompatibilität geprüft.
5. Der Name des importierten Parameters wird zur Parameterliste der entsprechenden Benutzerschnittstelle hinzugefügt.

**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.
- Die Parameterlisten wurden auf Default-Werte gesetzt.
- Die Parameterlisten der Benutzerschnittstellen wurden aktualisiert.

*Main Success Scenario:*

1.

**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.
- Die Parameterlisten wurden auf Default-Werte gesetzt.
- Die Parameterlisten der Benutzerschnittstellen wurden aktualisiert.

*Main Success Scenario:*

1.

**UC5: Select Graph**

| | |
|---|---|
| *Precondition:* | Die Benutzerschnittstelle zur Wahl eines Graphen ist aktiv. |

*Postconditions (success guarantee):*

- Der Graph wurde als ausgewählt markiert (*selected*).
- 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 (*enable*).
- Die Parameterlisten der Benutzerschnittstellen wurden aktualisiert.

*Main Success Scenario:*

1. In der Parameterliste wird der gewählte Graph alsaktuell gesetzt.
2. Der vormalige Graph wird im System entladen.
3. Der gewählte Graph wird im System geladen.
4. Die auf den Graphen anwendbaren Algorithmen 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.
- In der Benutzerschnittstelle zur Wahl des Algorithm steht mindestens ein aktivierter Algorithm zur Auswahl zur Verfügung.

*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 User wählt zur Berechnung der Traversal evtl. einen Start-, evtl. auch einen Endknoten.
5. Der gewählte Algorithm brechnet die Traversal.
6. Der User erhält eine Mitteilung, dass die Berechnung der Traversal abgeschlossen ist.
7. Der User quittiert die Mitteilung.
8. 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.

## 1.4. Glossary

**Algorithm** – Algorithmus: Anleitung, wie ein Graph durchschritten werden soll.

**Traversal** – Traversierung: Durchführung eines Algorithmus und Sammlung der Traversierungsschritte als Resultat. Der Graph erfährt bei jedem Traversierungsschritt eine Änderung seines Zustandes.

**Step** – Einzelner Schritt der Traversierung.

**Image** – Bild: Darstellung eines Zustandes eines Graphen.

**Step-by-Step** – Visualisierung, bei der ein Bild-Wechsel (Image) durch eine Benutzerinteraktion erfolgt.

**Steplength** – Anzahl Schritte (Steps) pro Bild-Wechsel (Image).

**Animation** – Visualisierung, bei der der Bild-Wechsel (Image) wiederholt automatisch, also ohne Benutzerinteraktion erfolgt.

**Delay** – Für die Animation: Verstreichende Zeit zwischen einem Bild-Wechsel (Image).

# 2. Design

## 2.1. Domain Model

### 2.1.1. Domain Model Diagram

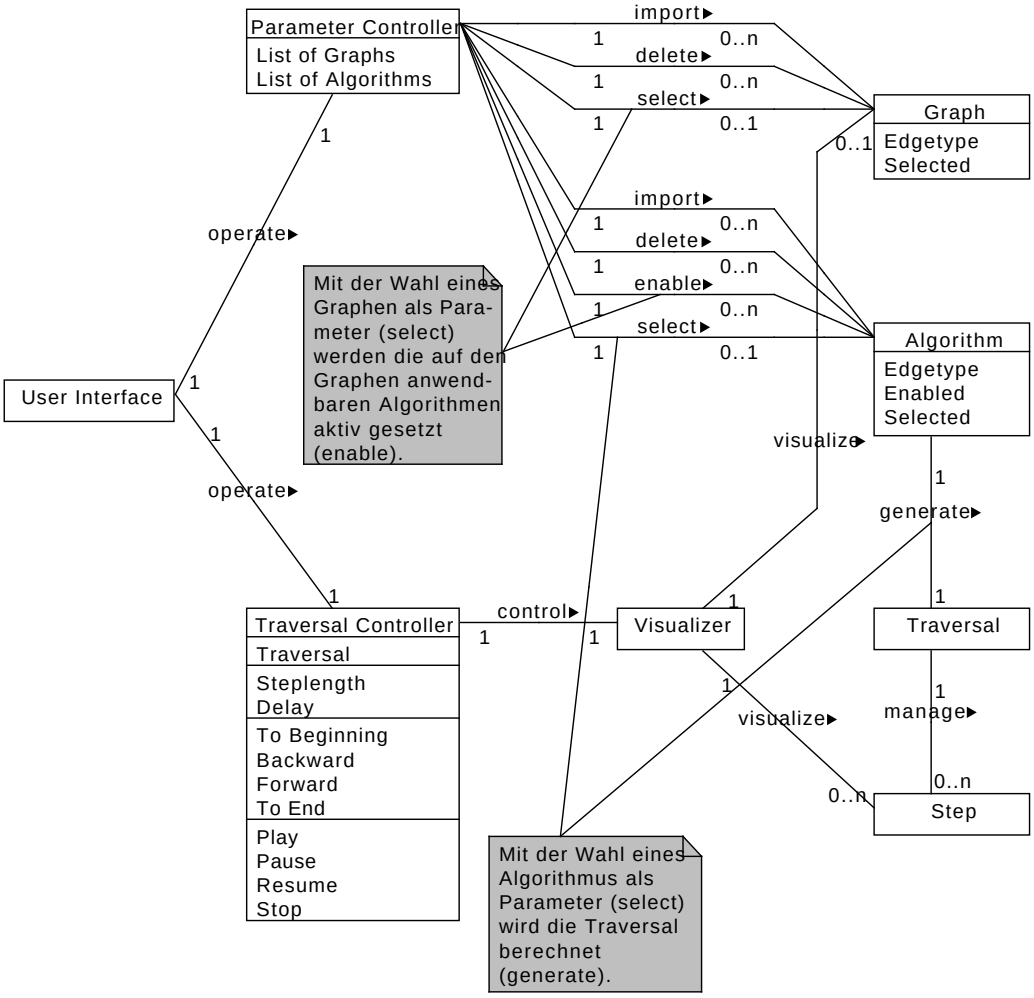


Figure 2.1.: Domain Model Diagram

### 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

Zur Elaboration des Designs werden zuerst die zwei Konzeptklassen *Parameter Controller* und *Traversal Controller* als State Machine (SM) betrachtet und je mit einem State Diagram illustriert.

Anschliessend 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.

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

Ein Controller implementiert ausschliesslich System-Operationen (und eventuell einige private Methoden). Jeder System-Event wird mit einer einzigen System-Operation eines Controllers assoziiert.

### 2.2.1. Parameter Controller

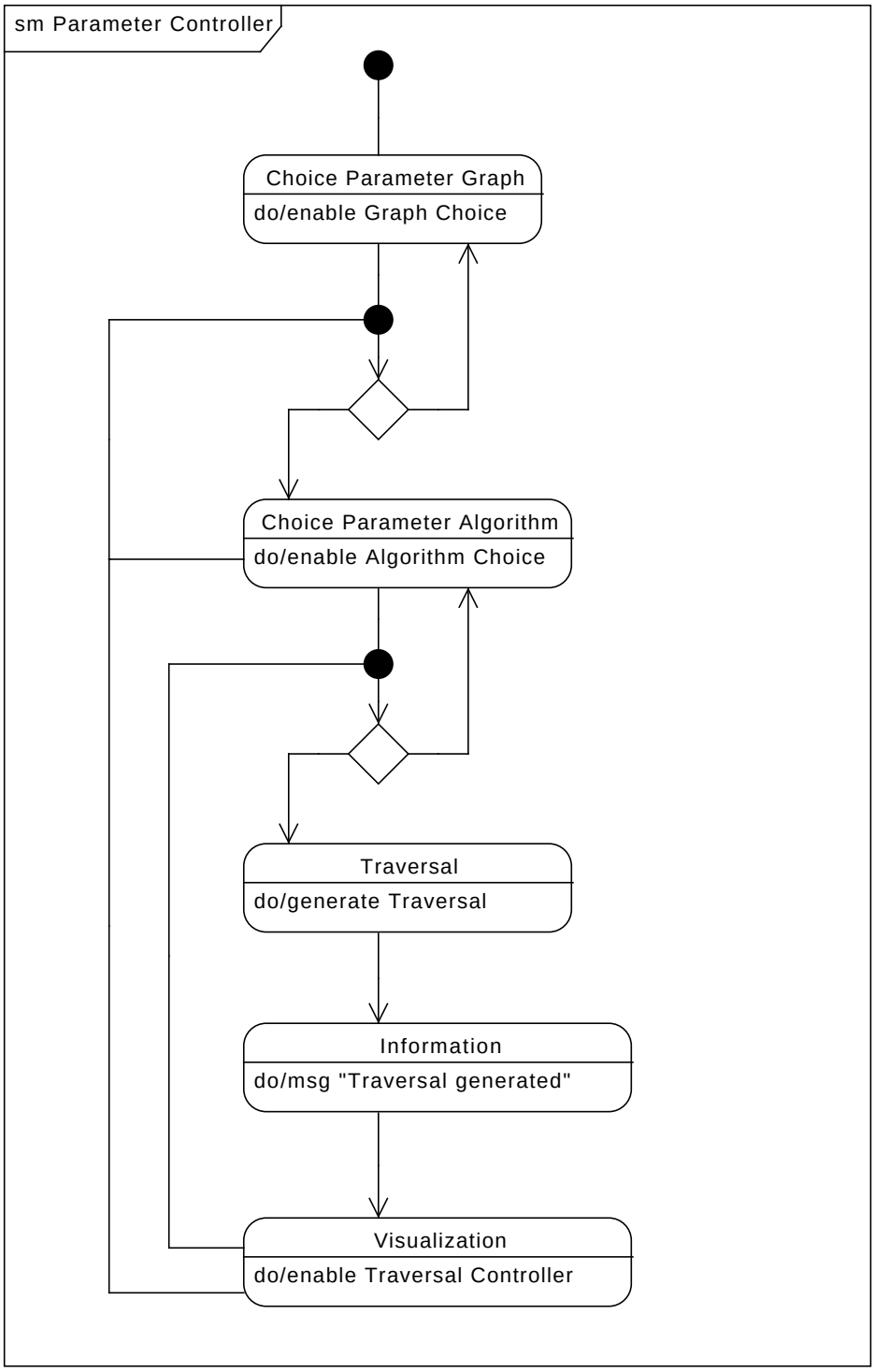


Figure 2.2.: Parameter Controller, State Diagram

### 2.2.2. Traversal Controller

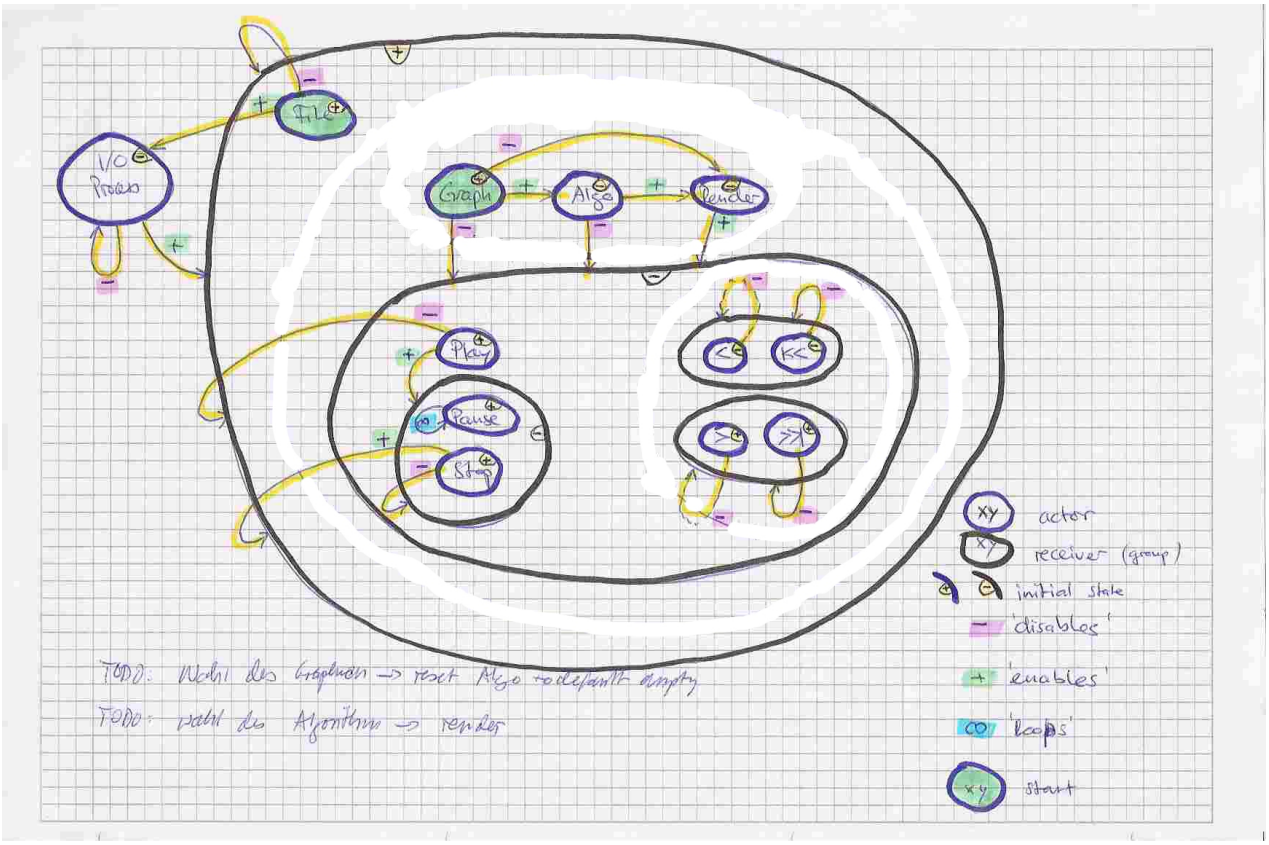


Figure 2.3.: Traversal Controller, State Diagram

### 2.2.3. UC1 Import Graph

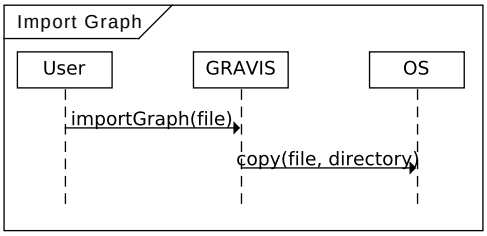


Figure 2.4.: UC1 Import Graph, System Sequence Diagram

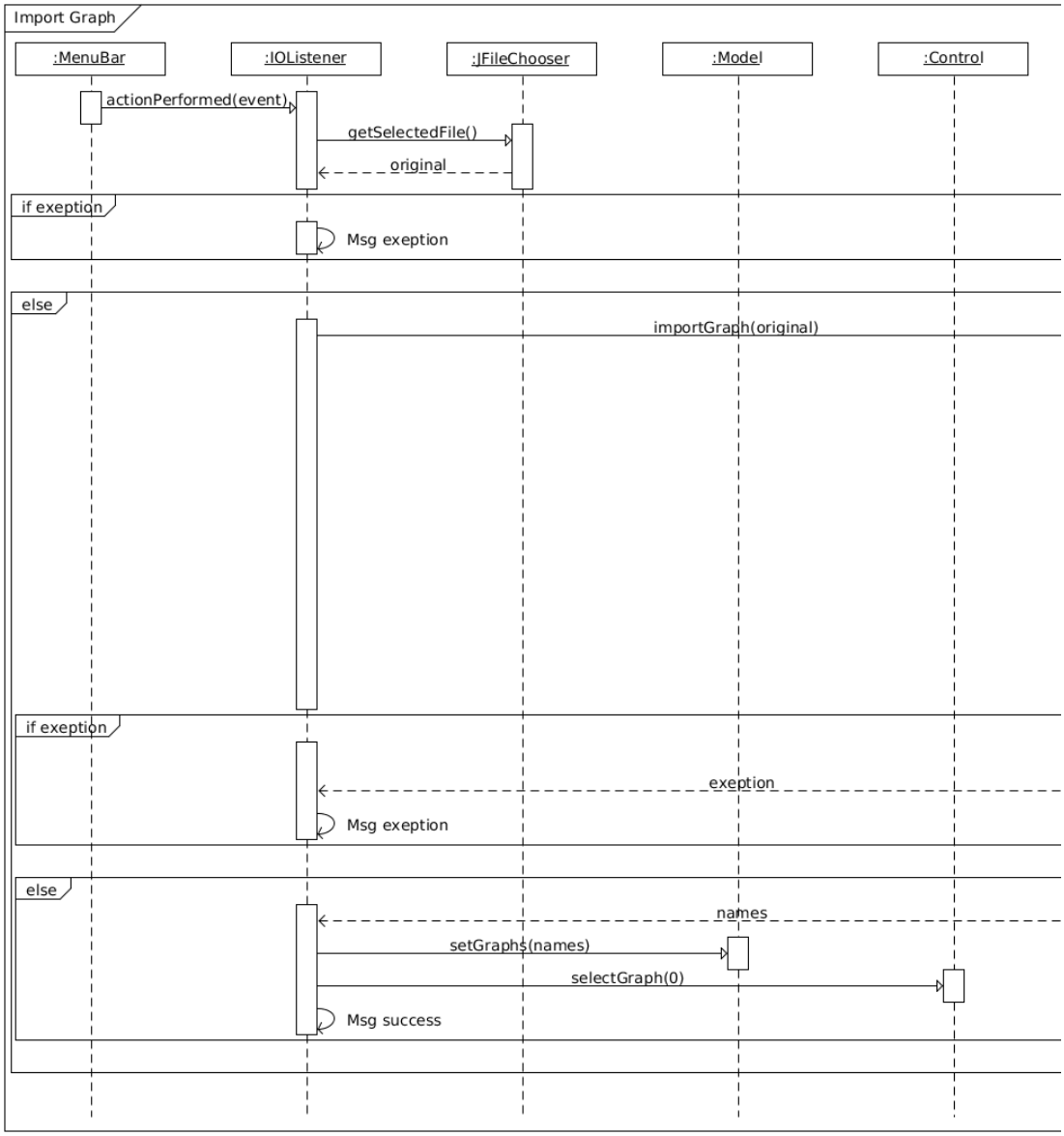


Figure 2.5.: UC1 Import Graph, Sequence Diagram 1/2

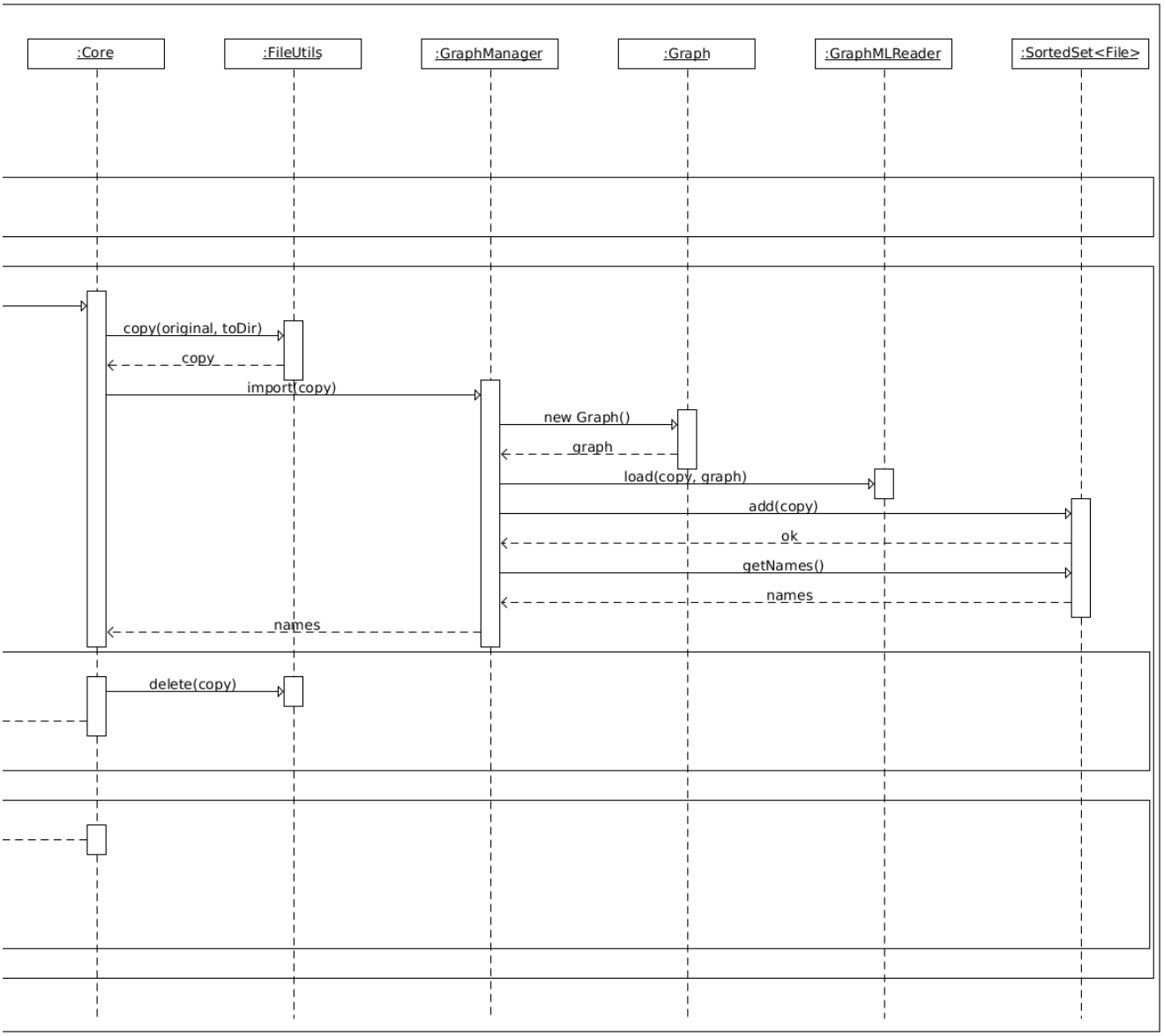


Figure 2.6.: UC1 Import Graph, Sequence Diagram 2/2

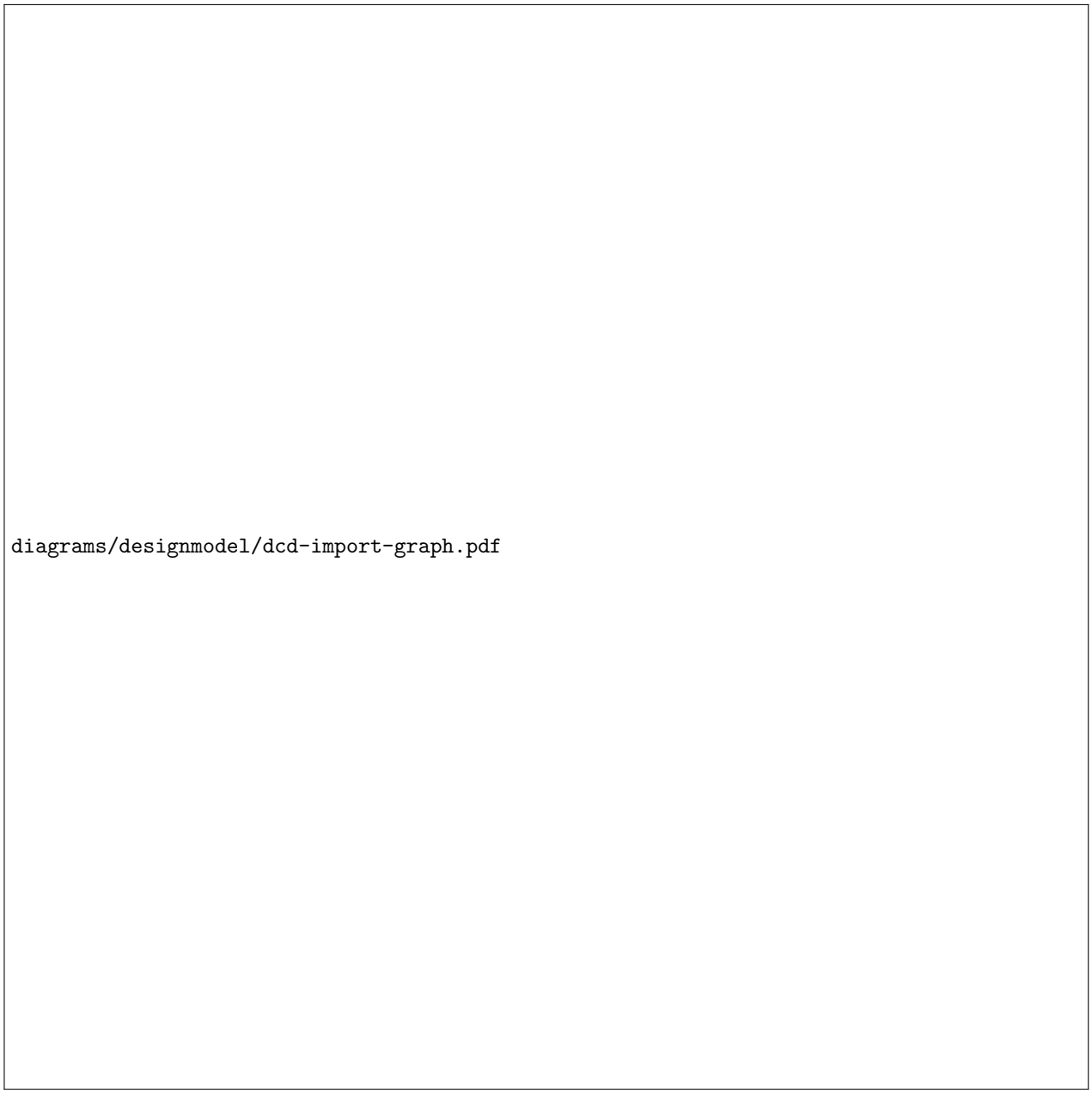


Figure 2.7.: UC1 Import Graph, Design Class Diagram

### 2.2.4. UC2 Import Algorithm

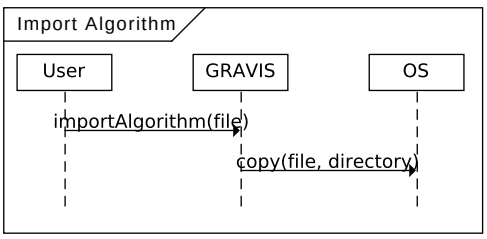


Figure 2.8.: UC2 Import Algorithm, System Sequence Diagram

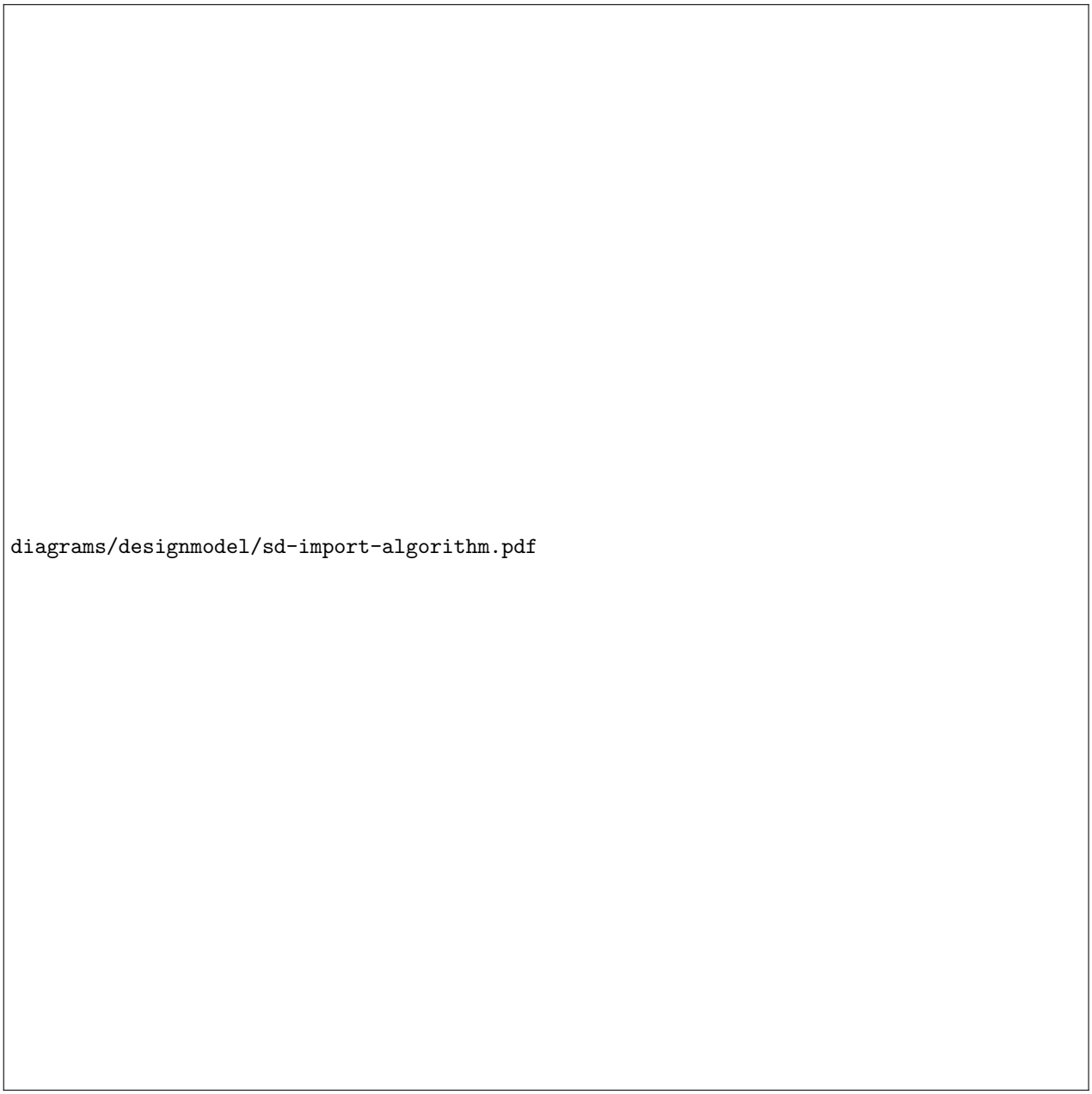


Figure 2.9.: UC2 Import Algorithm, Sequence Diagram

diagrams/designmodel/dcd-import-algorithm.pdf

Figure 2.10.: UC2 Import Algorithm, Design Class Diagram

### 2.2.5. UC3 Delete Graph

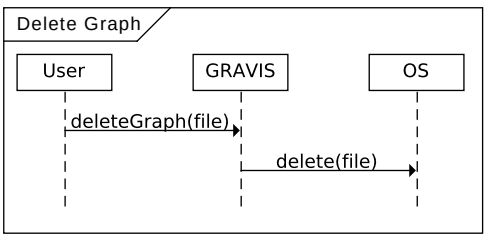


Figure 2.11.: UC3 Delete Graph, System Sequence Diagram

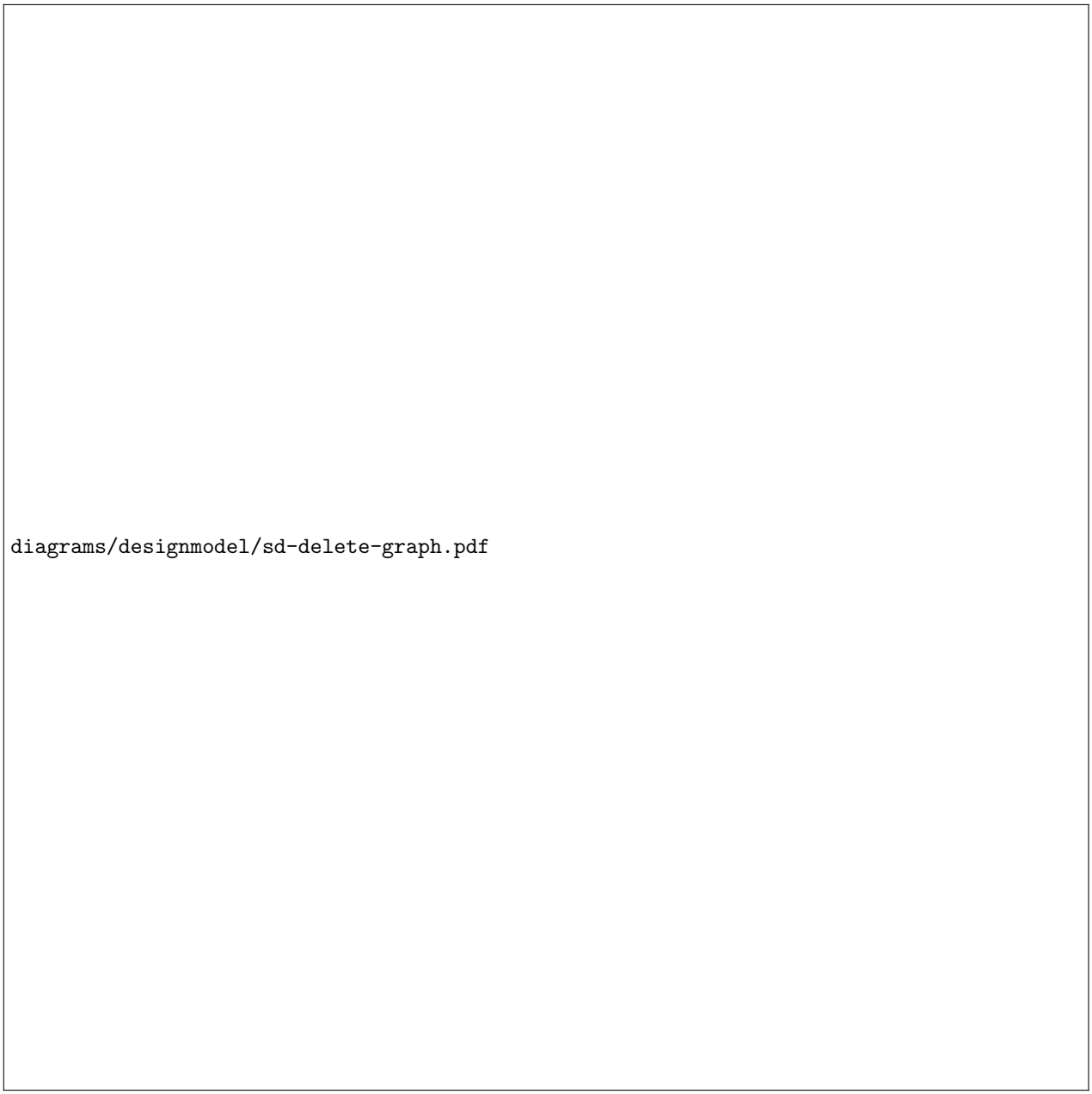


Figure 2.12.: UC3 Delete Graph, Sequence Diagram

diagrams/designmodel/dcd-delete-graph.pdf

Figure 2.13.: UC3 Delete Graph, Design Class Diagram

### 2.2.6. UC4 Delete Algorithm

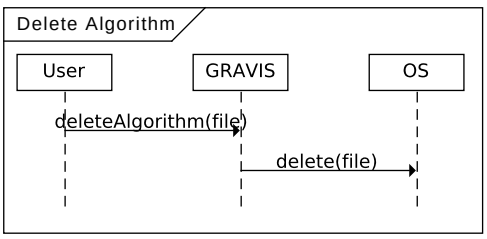


Figure 2.14.: UC4 Delete Algorithm, System Sequence Diagram

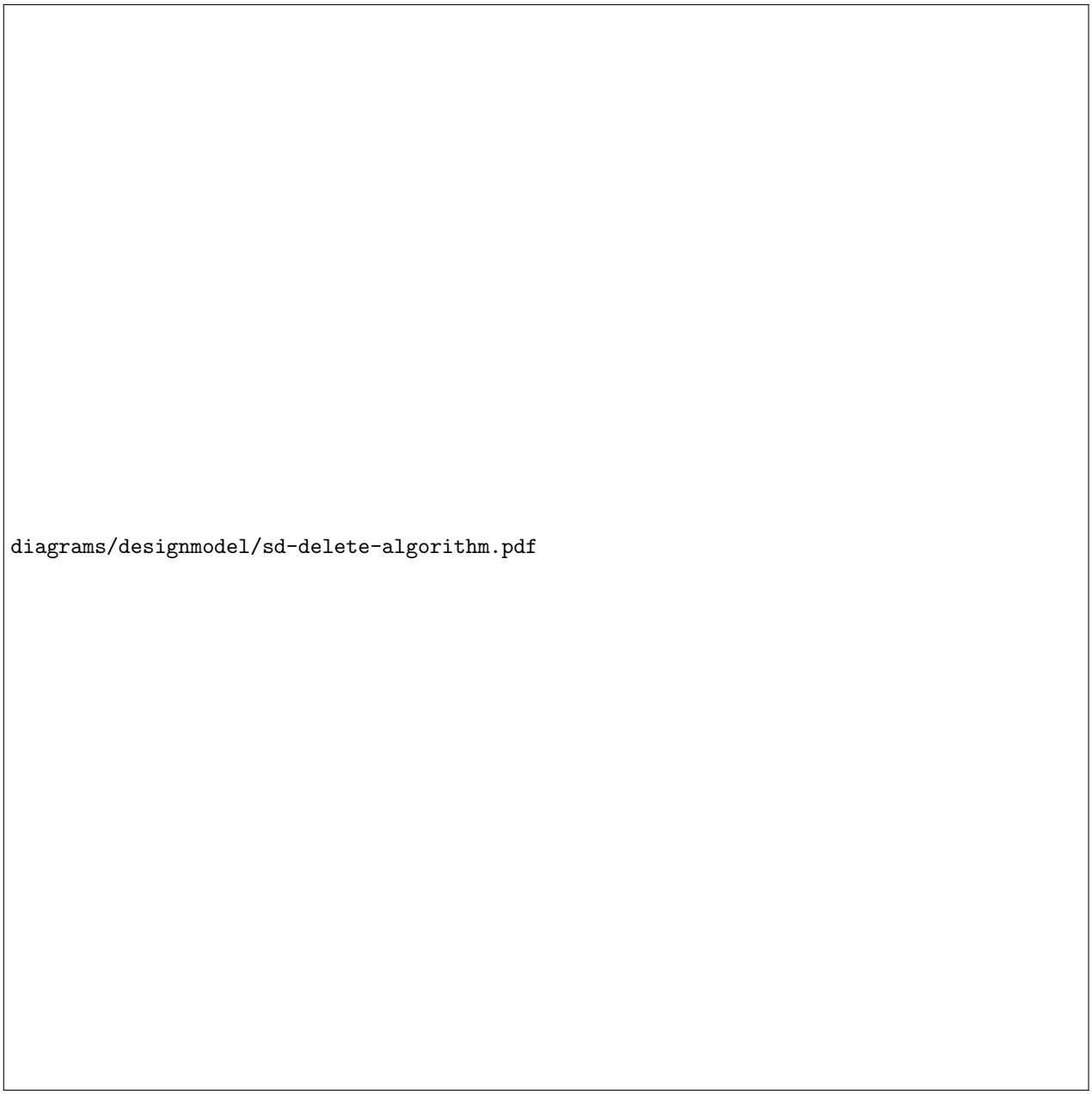


Figure 2.15.: UC4 Delete Algorithm, Sequence Diagram

diagrams/designmodel/dcd-delete-algorithm.pdf

Figure 2.16.: UC4 Delete Algorithm, Design Class Diagram

### 2.2.7. UC5 Select Graph

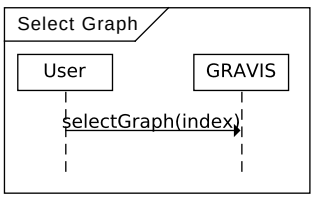


Figure 2.17.: UC5 Select Graph, System Sequence Diagram

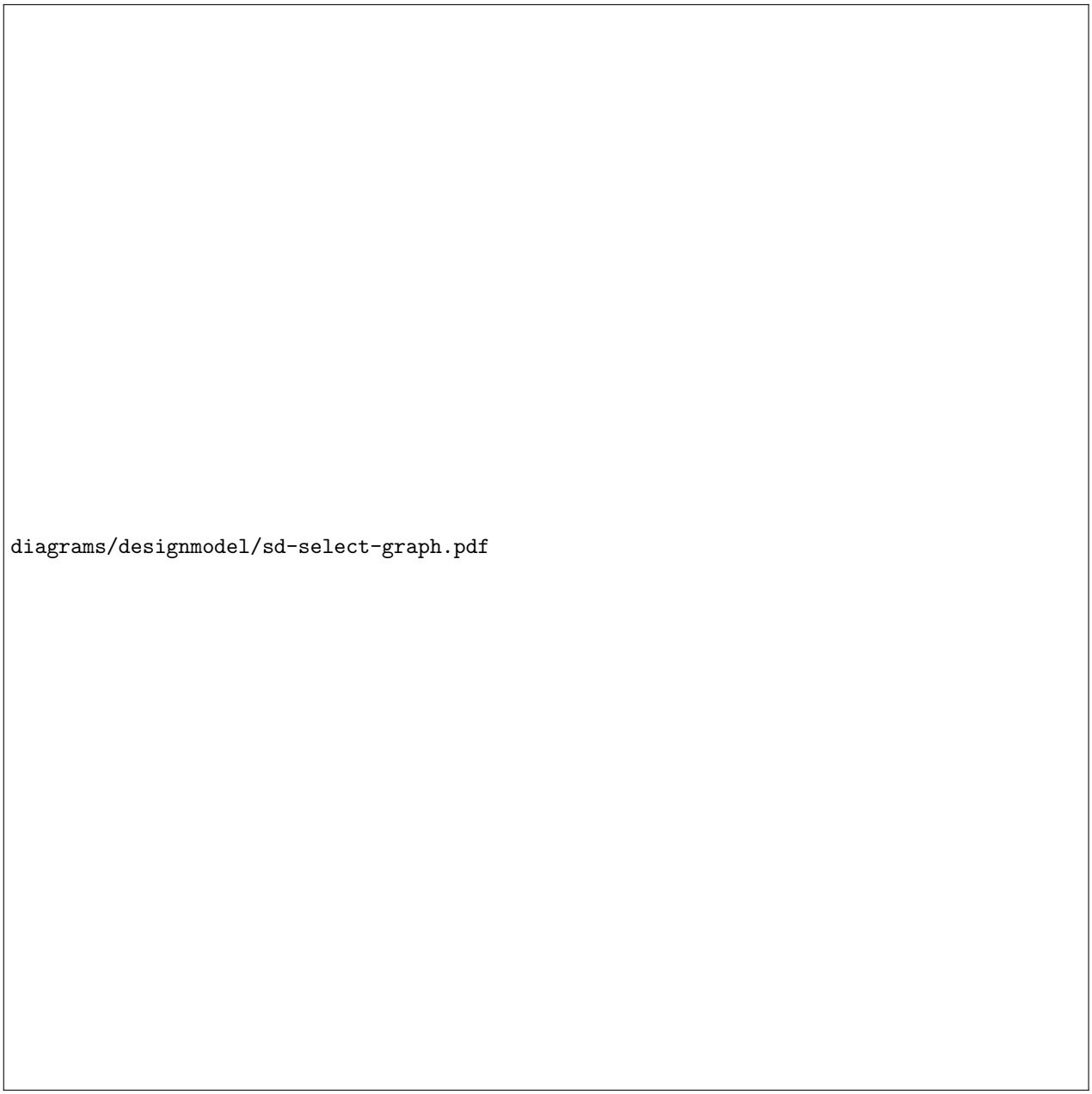


Figure 2.18.: UC5 Select Graph, Sequence Diagram

diagrams/designmodel/dcd-select-graph.pdf

Figure 2.19.: UC5 Select Graph, Design Class Diagram

### 2.2.8. UC6 Select Algorithm

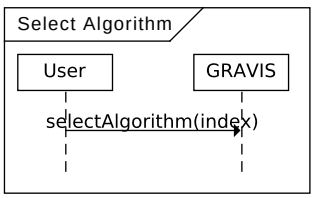


Figure 2.20.: UC6 Select Algorithm, System Sequence Diagram

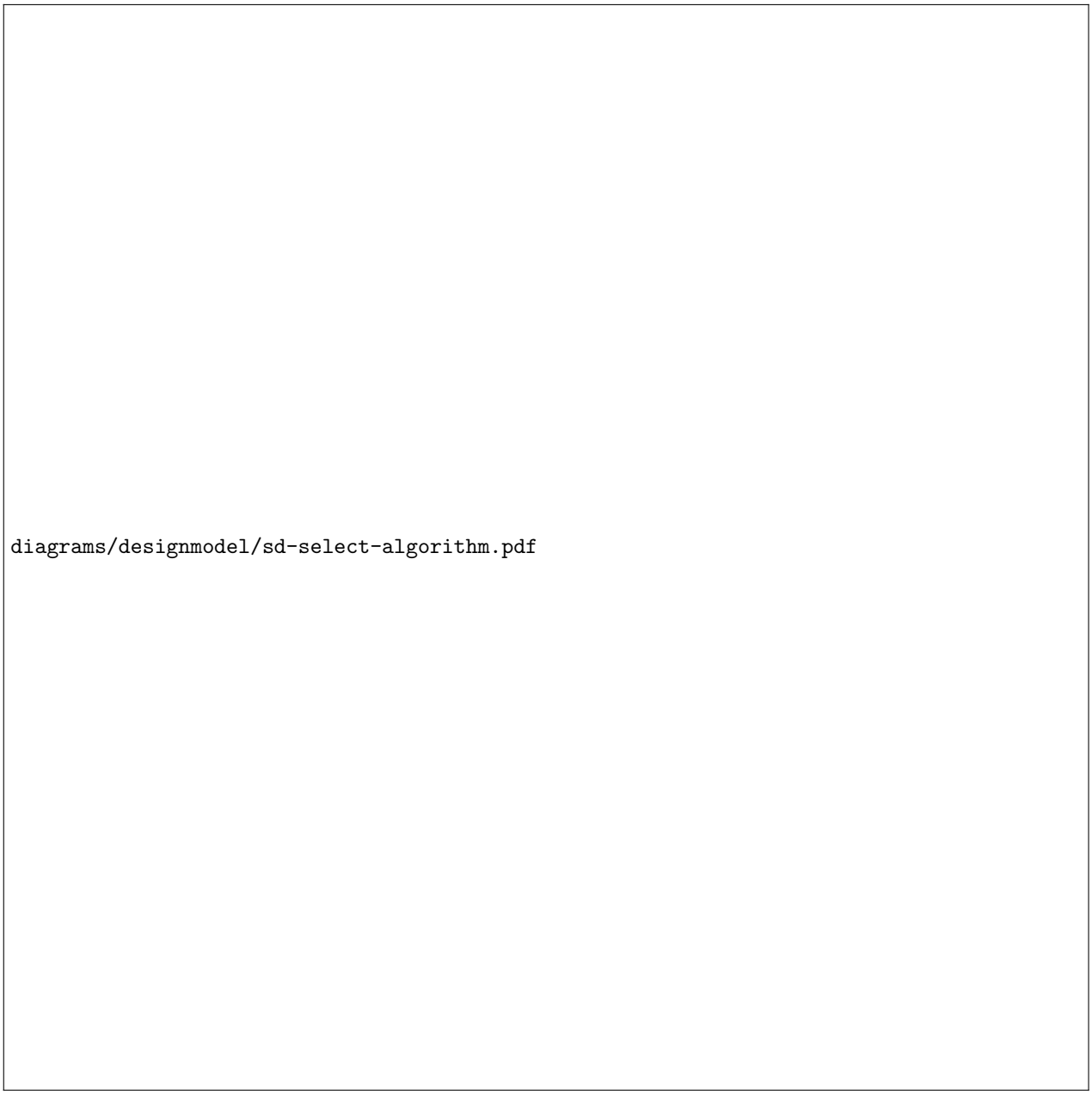


Figure 2.21.: UC6 Select Algorithm, Sequence Diagram


Figure 2.22.: UC6 Select Algorithm, Design Class Diagram

## 2.3. Software Architecture Document

### 2.3.1. Architektonische Entscheidungen

- 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. Logical View

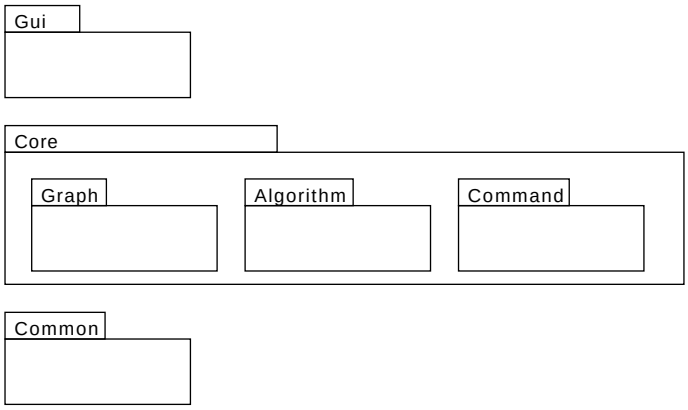


Figure 2.23.: Software Architecture, Package Diagram

**Beschreibung und Motivation**

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 hat sich aus den Requirements ergeben. Die Komponente Common hält die Interfaces für einen Actor 'Algorithm Developer' (offstage) bereit, die Komponente Gui enthält die (grafische) Benutzerschnittstelle und schliesslich die Komponente Core: das Herzstück des Systems kümmert sich um I/O, Verwaltung der Parameter sowie der Berechnung der Traversierung.

### 2.3.3. Process View

**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 sämtlichen GUI-Attributen und deren Getter- und Setter-Methoden
- View: Observer mit grafischen Elementen wie z.B. Menubar, Knöpfe, Regler und Text-Panelen
- Control: Implementiert Listeners und deren Methoden

### 2.3.4. 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

# 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
- Review, Anpassung der Planung

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

- Refactoring und Systemtests
- 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. Development Description

Als integrierte Entwicklungsumgebung wird die Software *Eclipse IDE* eingesetzt.

### 3.2.1. Programming Language and Libraries

Das System wird in der Programmiersprache Java implementiert. Es werden das Java Universal Network/Graph Framework JUNG [4], das XML-basierte Format GraphML [3], die Apache Commons IO Library und das Java Swing Framework verwendet.

### 3.2.2. Sourcecode Management

Für das Sourcecode Management (SCM) wird die Software git verwendet, das Projekt wird auf der webbasierten Plattform *github* gehalten.

Seit dem Entscheid in der Woche 45, das Projekt mit Maven zu erweitern, ist das aktuelle Projekt unter `https://github.com/brugr9/gravis` zu finden.

Das Vorgänger-Projekt ist weiterhin bis mindestens nach Erhalt der Modul-Bewertung erreichbar über den URL `https://github.com/brugr9/ch.bfh.bti7301.hs2013.GraphVisualisierung2`.

### 3.2.3. 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 [2]. Für die Formulierung der Use Cases im ausgearbeiteten Format wurde ein LaTeX-Style-File [1] erstellt und verwendet.

# A. Appendix

# List of Figures

# List of Tables

# Bibliography


[1] R. E. Bruggmann. ooaduc.sty - LaTeX-Style File for Use Cases in Object-Oriented Analysis and Design. Accessed 20-October-2013. [Online]. Available: https://gist.github.com/brugr9/5178487

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

[3] G. Team. The GraphML File Format. Accessed 19-October-2013. [Online]. Available: http://graphml.graphdrawing.org/

[4] T. J. F. D. Team. JUNG, Java Universal Network/Graph Framework. Accessed 19-October-2013. [Online]. Available: http://jung.sourceforge.net/