Berner Fachhochschule
Technik und Informatik

Projekt 1

Modul BTI-7301, Herbstsemester 2013/14

Graph Visualisierung 2 (GRAVIS)

Object Oriented Analysis and Design (OOAD), Revision 1.5

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

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

Datum: 6. November 2013

Desktop-Programm zur visualisierten Traversierung von Graphen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

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 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.

1.2 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, einfach- oder mehrfachkantig).
- 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.

2 Project management

2.1 Sourcecode management

Für das Sourcecode management (SCM) wird die Software git verwendet, das Projekt wird auf github gehalten (`https://github.com/brugr9/ch.bfh.bti7301.hs2013.GraphVisualisierung2`).

2.2 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)

- Einarbeitung 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 Architecture

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.

3.1 Common

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

3.2 Core

Die Komponente implementiert:

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

Es werden das Java Universal Network/Graph Framework (JUNG, `http://jung.sourceforge.net/`) sowie das XML-basierte Format GraphML (`http://graphml.graphdrawing.org/`) verwendet.

3.3 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

Es wird das Java Swing Framework verwendet.

Gui Elemente

- Daten:
  - Neuen Graphen oder Algorithmus importieren
  - Importierter Graph oder Algorithmus löschen
- Traversierung:
  - Graph auswählen
  - Algorithmus auswählen
  - evt. Start- resp. Endknoten auswählen
- Visualisierung:

- Einstellung Step: Anzahl Traversierungs-Schritte pro Bild
- Einstellung 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
- Anzeige des Visualisierungsfortschrittes in einer Progressbar

4 Use Cases Model

4.1 Use Cases Diagram

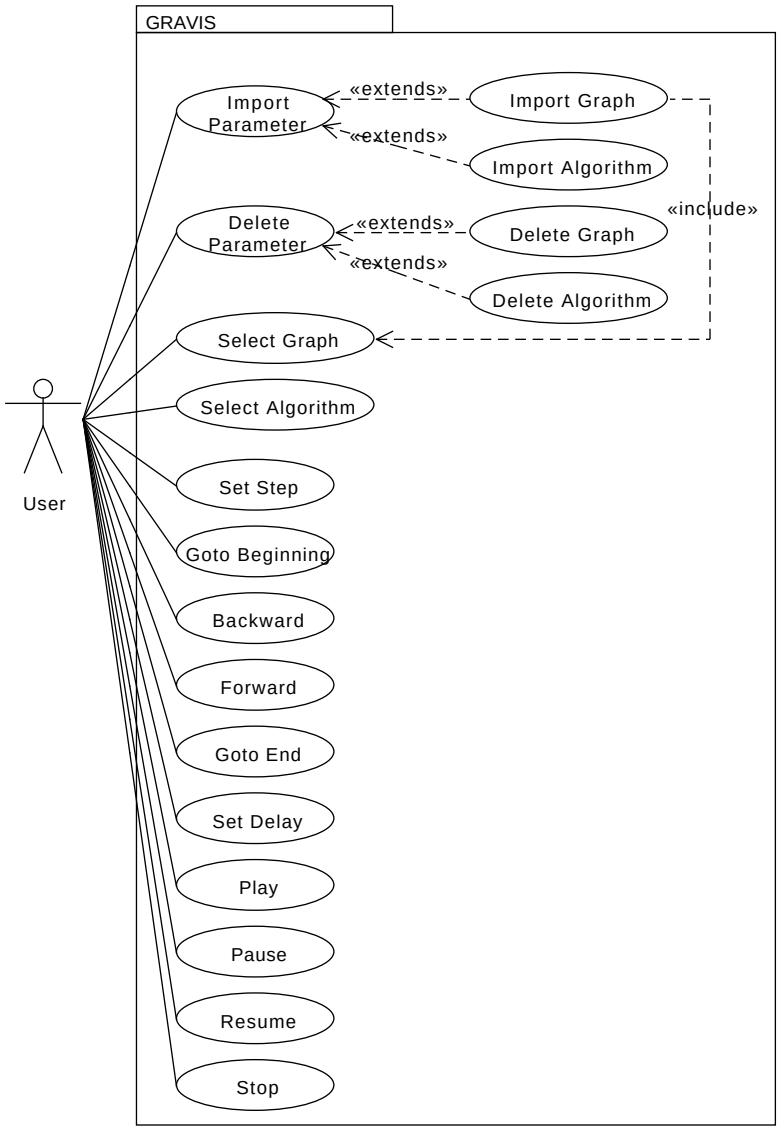

Abbildung 1: Use Cases Diagram

4.2 Use Cases in brief format

Import Graph: Der Benutzer kann einen neuen Graphen importieren.

(Ausgearbeitetes Format siehe Seite 8)

Import Algorithm: Der Benutzer kann einen neuen Algorithmus importieren.

(Ausgearbeitetes Format siehe Seite 10)

Delete Graph: Der Benutzer kann einen importierten Graphen löschen.

(Ausgearbeitetes Format siehe Seite 12)

Delete Algorithm: Der Benutzer kann einen importierten Algorithmus löschen.

(Ausgearbeitetes Format siehe Seite 13)

Select Graph: Der Benutzer kann einen Graphen auswählen.

(Ausgearbeitetes Format siehe Seite 14)

Select Algorithm: Der Benutzer kann einen Algorithmus auswählen, evt. Start- resp. Endknoten auswählen.

(Ausgearbeitetes Format siehe Seite 15)

Set Step: Der Benutzer kann für die Visualisierung die Anzahl Traversierungsschritte pro Bild einstellen.

(Ausgearbeitetes Format siehe Seite 16)

Forward: Der Benutzer kann in der Visualisierung ein Bild vorwärts gehen.

(Ausgearbeitetes Format siehe Seite 17)

Backward: Der Benutzer kann in der Visualisierung ein Bild rückwärts gehen.

(Ausgearbeitetes Format siehe Seite 18)

Goto Beginning: Der Benutzer kann in der Visualisierung an das Ende springen.

(Ausgearbeitetes Format siehe Seite 19)

Goto End: Der Benutzer kann in der Visualisierung an den Anfang springen.

(Ausgearbeitetes Format siehe Seite 20)

Set Delay: Der Benutzer kann für die Visualisierung das Zeitintervall zwischen zwei Bildern einstellen.

(Ausgearbeitetes Format siehe Seite 21)

Play: Der Benutzer kann das Streaming der Visualisierung starten.

(Ausgearbeitetes Format siehe Seite 22)

Pause: Der Benutzer kann das Streaming der Visualisierung anhalten.

(Ausgearbeitetes Format siehe Seite 23)

Resume: Der Benutzer kann das Streaming der Visualisierung wiederaufnehmen.

(Ausgearbeitetes Format siehe Seite 24)

Stop: Der Benutzer kann das Streaming der Visualisierung stoppen.

(Ausgearbeitetes Format siehe Seite 25)

4.3 Use Cases in fully dressed format

Für alle Use Cases gilt:

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

Import Graph

UC1: Import Graph

Preconditions:

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

Postconditions (success guarantee):

- Der Graph steht dem System als Parameter zur weiteren Verarbeitung zur Verfügung.
- Der Graph steht dem User in der Graph-Parameterliste zur Auswahl bereit.
- Der Graph wurde durch das System ausgewählt und im GUI dargestellt.

Main Success Scenario:

1. Der Benutzer startet den **Graph-Import**.
2. Der Benutzer 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 Name des importierten Graphen wird zur Graph-**Parameterliste** hinzugefügt.
6. Der Graph wird durch das System in der Graph-Parameterliste **ausgewählt** (siehe *UC Select Graph*, Seite 14.

Extensions (Alternative Flows):

1.a
1. Das Starten des Vorganges **schlägt fehl**.
2. Eine Fehlermeldung wird ausgegeben.

2.a
1. Der Benutzer **bricht den Vorgang ab**.

2.b
1. Die angegebene Datei kann **nicht gefunden** werden.
2. Eine Fehlermeldung wird ausgegeben.
3. Dem Benutzer wird widerum die Möglichkeit gegeben, den Pfad und den Dateinamen einer Datei anzugeben oder den Vorgang abzubrechen (Rekursion).

3.a
1. Die angegebene Datei **kann nicht** in die Dateistruktur des Systems **kopiert werden**.
2. Eine Fehlermeldung wird ausgegeben.

4.a
1. Die angegebene Datei ist **nicht kompatibel**.
2. Die kopierte Datei wird aus der Dateistruktur des Systems gelöscht.
3. Eine Fehlermeldung wird ausgegeben.

5.a
1. Der Name des importierten Graphen **kann nicht zur Parameterliste hinzugefügt werden**.
2. Die kopierte Datei wird aus der Dateistruktur des Systems gelöscht.
3. Eine Fehlermeldung wird ausgegeben.

6.a
1. siehe *UC Select Graph*, Seite 14 resp. *UC Select Algorithm*, Seite 15.

Import Algorithm

UC2: Import Algorithm	
Preconditions: • Die Dateistruktur des Betriebssystems ist zugänglich. • Auf die zu importierende Datei sind mindestens Leserechte gesetzt.	
Postcondition (success guarantee):	Der Algorithmus steht dem System als Parameter zur weiteren Verarbeitung zur Verfügung.
Main Success Scenario: 1. Der Benutzer startet den **Algorithmus-Import**. 2. Der Benutzer 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 Name des importierten Parameters wird zur **Parameterliste** der entsprechenden Benutzerschnittstelle hinzugefügt.	

Extensions (Alternative Flows):

1.a
1. Das Starten des Vorganges **schlägt fehl**.
2. Eine Fehlermeldung wird ausgegeben.

2.a
1. Der Benutzer **bricht den Vorgang ab**.

2.b
1. Die angegebene Datei kann **nicht gefunden** werden.
2. Eine Fehlermeldung wird ausgegeben.
3. Dem Benutzer wird widerum die Möglichkeit gegeben, den Pfad und den Dateinamen einer Datei anzugeben oder den Vorgang abzubrechen (Rekursion).

3.a
1. Die angegebene Datei **kann nicht** in die Dateistruktur des Systems **kopiert werden**.
2. Eine Fehlermeldung wird ausgegeben.

4.a
1. Die angegebene Datei ist **nicht kompatibel**.
2. Die Datei wird aus der Dateistruktur des Systems gelöscht.
3. Eine Fehlermeldung wird ausgegeben.

5.a
1. Der Name des importierten Parameters **kann nicht zur Parameterliste hinzugefügt werden**.
2. Der importierte Parameter wird aus dem Arbeitsspeicher gelöscht.
3. Die importierte Datei wird aus der Dateistruktur des Systems gelöscht.
4. Eine Fehlermeldung wird ausgegeben.

6.a
1. siehe *UC Select Graph*, Seite 14 resp. *UC Select Algorithm*, Seite 15.

Delete Graph

UC3: Delete Graph	
Precondition:	Der zu löschende Graph steht als **Datei** zur weiteren Verarbeitung zur Verfügung.

Postconditions (success guarantee):

- Die Graph-Datei wurde aus dem System **gelöscht**.
- Die Parameterliste wurde **aktualisiert**.

Main Success Scenario:

1.

Extensions (Alternative Flows):

1.a 1.

Delete Algorithm

UC4: Delete Algorithm	
Precondition:	Der zu löschende Algorithmus steht als **Datei** zur weiteren Verarbeitung zur Verfügung.

Postconditions (success guarantee):

- Die Algorithmus-Datei wurde aus dem System **gelöscht**.
- Die Parameterliste wurde **aktualisiert**.

Main Success Scenario:

1.

Extensions (Alternative Flows):

1.a 1.

Select Graph

UC5: Select Graph

Preconditions:

- Die Benutzerschnittstelle zur Wahl des Graphen ist aktiv.

Postconditions (success guarantee):

- Der gewählte Graph steht als **geladene Instanz** zur weiteren Verarbeitung zur Verfügung.
- Der gewählte Graph ist visualisiert.

Main Success Scenario:

1. In der Parameterliste wird der gewählte Graph als**aktuell** gesetzt.
2. Der vormalige Graph wird im System **entladen**.
3. Der gewählte Graph wird im System **geladen**.
4. Die Parameterliste zur Wahl des Algorithmus wird **angepasst**.
5. Als Visualisierung wird der aktuelle Graph gezeichnet.

Extensions (Alternative Flows):

1.a
1. Das Setzen des gewählten Graphen als aktuell **schlägt fehl**.
2. Eine Fehlermeldung wird ausgegeben.

2.a
1. Der vormalige Graph kann **nicht entladen** werden.
2. In der Parameterliste wird der vormalige Graph als aktuell gesetzt.
3. Eine Fehlermeldung wird ausgegeben.

3.a
1. Der gewählte Graph kann **nicht geladen** werden.
2. In der Parameterliste Graph wird ein leerer Default-Wert als aktuell gesetzt.
3. Eine Fehlermeldung wird ausgegeben.

Select Algorithm

UC6: Select Algorithm

Preconditions:

- Die Benutzerschnittstelle zur Wahl des Algorithmus ist aktiv.

Postconditions (success guarantee):

- Der gewählte Algorithmus steht als **geladene Instanz** zur weiteren Verarbeitung zur Verfügung.

Main Success Scenario:

1. In der Parameterliste wird der gewählte Algorithmus als **aktuell** gesetzt.
2. Der vormalige Algorithmus wird im System **entladen**.
3. Der gewählte Algorithmus wird im System **geladen**.

Extensions (Alternative Flows):

1.a
1. Das Setzen des gewählten Algorithmus als aktuell **schlägt fehl**.
2. Eine Fehlermeldung wird ausgegeben.

2.a
1. Der vormalige Algorithmus kann **nicht entladen** werden.
2. In der Parameterliste wird der vormalige Algorithmus als aktuell gesetzt.
3. Eine Fehlermeldung wird ausgegeben.

3.a
1. Der gewählte Algorithmus kann **nicht geladen** werden.
2. In der Parameterliste wird ein leerer Default-Wert als aktuell gesetzt.
3. Eine Fehlermeldung wird ausgegeben.

Set Step

UC7: Set Step
Precondition:
Postconditions (success guarantee): • •
Main Success Scenario: 1.
Extensions (Alternative Flows): 1.a 1.

Forward

UC8: Forward
Precondition:
Postconditions (success guarantee): • •
Main Success Scenario: 1.
Extensions (Alternative Flows): 1.a 1.

Backward

UC9: Backward
Precondition:
Postconditions (success guarantee): • •
Main Success Scenario: 1.
Extensions (Alternative Flows): 1.a 1.

Goto Beginning

UC10: Goto Beginning
Precondition:
Postconditions (success guarantee): • •
Main Success Scenario: 1.
Extensions (Alternative Flows): 1.a 1.

Goto End

UC11: Goto End
Precondition:
Postconditions (success guarantee): • •
Main Success Scenario: 1.
Extensions (Alternative Flows): 1.a 1.

Set Delay

UC12: Set Delay
Precondition:
Postconditions (success guarantee): • •
Main Success Scenario: 1.
Extensions (Alternative Flows): 1.a 1.

Play

UC13: Play
Precondition:
Postconditions (success guarantee): • •
Main Success Scenario: 1.
Extensions (Alternative Flows): 1.a 1.

Pause

UC14: Pause
Precondition:
Postconditions (success guarantee): • •
Main Success Scenario: 1.
Extensions (Alternative Flows): 1.a 1.

Resume

UC15: Resume
Precondition:
Postconditions (success guarantee): • •
Main Success Scenario: 1.
Extensions (Alternative Flows): 1.a 1.

Stop

UC16: Stop
Precondition:
Postconditions (success guarantee): • •
Main Success Scenario: 1.
Extensions (Alternative Flows): 1.a 1.

Der Visualisierungsfortschritt wird in der Progressbar angezeigt.

5 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.

6 Glossary

Algorithmus – Anleitung, wie eine Datenstruktur 'Graph' durchschritten werden soll.

Traversal – Durchführung eines Algorithmus. Dabei erfährt der Graph bei jedem Traversierungsschritt eine Änderung seines Zustandes, welche aufgelistet als Resultat der Traversierung gilt.

Step – Einzelner Schritt der Traversierung.

Frame – Bild, Darstellung eines Zustandes des Graphen.

Step-by-step – Eigentlich 'frame-by-frame': Visualisierung, bei der der Wechsel vom aktuellen zum nächsten Frame eine Benutzerinteraktion verlangt.

Animation – Visualisierung, bei der der Wechsel vom aktuellen zum nächsten Frame automatisch (ohne Benutzerinteraktion) erfolgt.

Delay – Für die Animation: Zeitintervall zwischen der Visualisierung von zwei Frames.

7 Domain Model

7.1 Domain Model Diagram

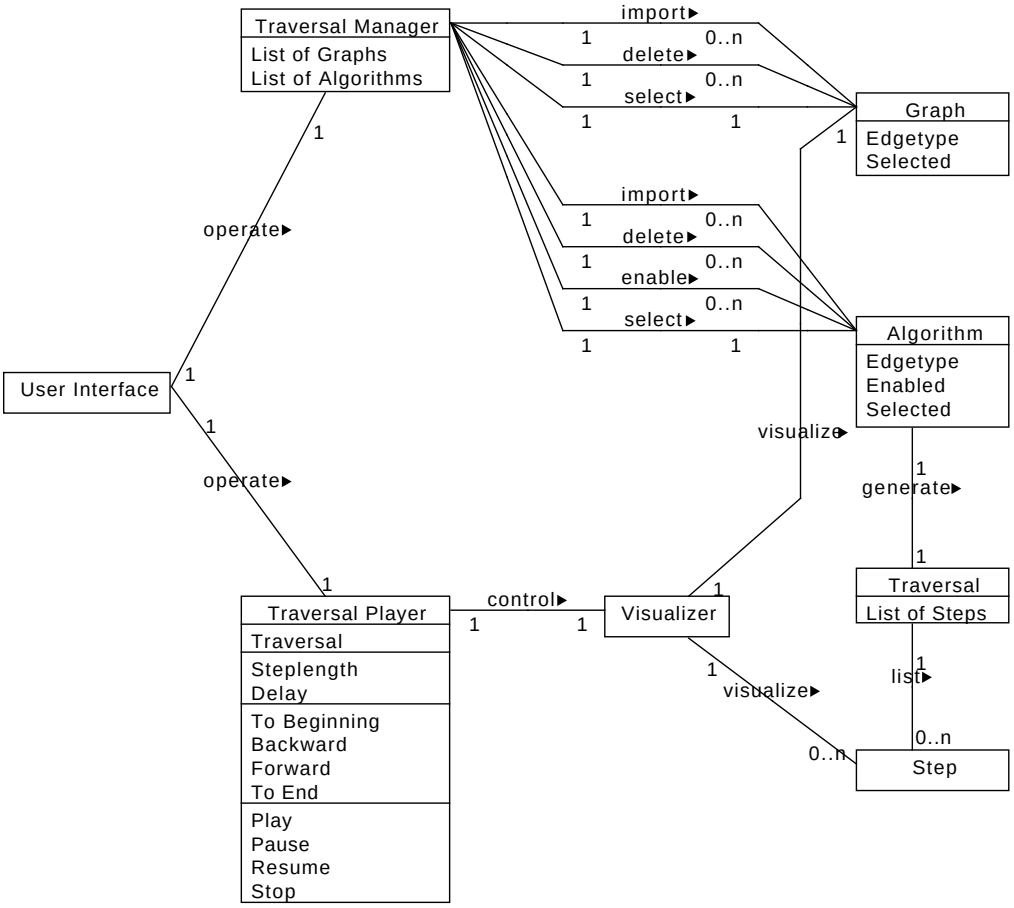

Abbildung 2: Domain Model Diagram

7.2 Domain Model Description

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

User interface

- Über ein User Interface (1) kann ein User einen Traversal Manager (1) bedienen (*operate*).
- Über ein User Interface (1) kann ein User einen Traversal Player (1) bedienen (*operate*).

Traversal Manager

- Ein Traversal Manager hält eine gegebene Anzahl Graphen als Vorlagen (*List of Graphs*) bereit.
- Über einen Parameter Manager (1) kann ein User einen oder mehrere Graphen (0..n) des Formates *.graphml importieren (*import*). Diese werden der Liste mit Graphen (*List of Graphs*) hinzugefügt.
- Über einen Parameter Manager (1) kann ein User vormals importierte Graphn (0..n) wieder löschen (*delete*). Diese werden aus der Liste mit Graphen (*List of Graphs*) wieder entfernt.
- Über einen Parameter Manager (1) kann ein User einen Graphen (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 Traversal Manager hält eine gegebene Anzahl Algorithmen als Vorlagen (*List of Algorithms*) bereit. Diese werden der Liste mit Algorithmen (*List of Algorithms*) hinzugefügt.
- Über einen Parameter Manager (1) kann ein User einen oder mehrere Algorithmen (0..n) importieren (*import*), sofern dieser die geforderten Interfaces implementiert.
- Über einen Parameter Manager (1) kann ein User vormals importierte Algorithmen (0..n) wieder löschen (*delete*). Diese werden aus der Liste mit Algorithmen (*List of Algorithms*) wieder entfernt.
- Über einen Parameter Manager (1) kann ein User einen Algorithmus (1) als Parameter auswählen (*select*).
- Mit der Wahl eines Algorithmus (1) als Parameter wird eine Traversal (1) erstellt (*generate*).

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 kann aktiviert werden (*Enabled*).
- Ein Algorithm kann ausgewählt werden (*Selected*).
- Ein Algorithm (1) generiert eine Traversal (1).

Traversal

- Eine Traversal (1) kann einen oder mehrere Steps (0..n) auflisten (*List of Steps*).

Step

- Ein Step ist ein Schritt der Traversierung.

Traversal Player

- Ein Traversal Player (1) steuert einen Visualizer (1).
- Ein Traversal Player hält eine Traversierung (*Traversal*).
- Über einen Traversal Player (1) kann ein User die Anzahl Schritte (*Steplength*) pro Bild einstellen.
- Über einen Traversal Player (1) kann ein User die Zeit (*Delay*) zwischen zwei Bildern einstellen.
- Über einen Traversal Player (1) kann ein User Springen-Elemente (*Forward*, *Backward*, *To Beginning*, *To End*) für die Step-by-Step Visualisierung im Visualizer (1) bedienen.
- Über einen Traversal Player (1) kann ein User Abspiel-Elemente (*Play*, *Pause*, *Resume*, *Stop*) für die animierte Visualisierung im Visualizer (1) bedienen.

Visualizer

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