|  |
| --- |
|  |
| Graph Visualizer  Eine Applikation zur Visualisierung von Graphen und Algorithmen  **Projekt-Dokumentation** |
| Studiengang: Informatik, Modul BTI7301 (Projekt 1), HS 2013/14  Autor: Patrick Kofmel ([kofmp1@bfh.ch](mailto:kofmp1@bfh.ch))  Betreuer: Jürgen Eckerle ([juergen.eckerle@bfh.ch](mailto:juergen.eckerle@bfh.ch))  Datum: 13. Juni 2014 |

Inhaltsverzeichnis

[1 Einleitung 3](#_Toc390603105)

[2 Requirements und Use Cases 3](#_Toc390603106)

[2.1 Projektbeschreibung 3](#_Toc390603107)

[2.2 System Requirements 3](#_Toc390603108)

[2.3 Use Cases 4](#_Toc390603109)

[3 Systemarchitektur und Design 4](#_Toc390603110)

[3.1 Schichtenarchitektur 4](#_Toc390603111)

[3.2 MVC-Pattern 4](#_Toc390603112)

[3.3 Knoten und Kanten 4](#_Toc390603113)

[3.4 Graph-Klassen 4](#_Toc390603114)

[3.5 Algorithmen-Klassen 4](#_Toc390603115)

[3.6 Step-Klassen 4](#_Toc390603116)

[4 Implementation und Tests 4](#_Toc390603117)

[4.1 Coding Conventions 4](#_Toc390603118)

[4.2 Algorithmen-Implementation 4](#_Toc390603119)

[4.2.1 Tiefensuche (DFS) 5](#_Toc390603120)

[4.2.2 Breitensuche (BFS) 5](#_Toc390603121)

[4.2.3 Dijkstra 5](#_Toc390603122)

[4.2.4 Kruskal 5](#_Toc390603123)

[4.3 Tests 5](#_Toc390603124)

[5 Benutzer-Dokumentation 5](#_Toc390603125)

[5.1 Programm 5](#_Toc390603126)

[5.2 Graphml-Format 5](#_Toc390603127)

[6 Abbildungsverzeichnis 6](#_Toc390603128)

[7 Tabellenverzeichnis 6](#_Toc390603129)

[8 Glossar 6](#_Toc390603130)

[9 Literaturverzeichnis 6](#_Toc390603131)

[10 Anhang 7](#_Toc390603132)

# Einleitung

Dieses Dokument enthält die Projekt-Dokumentation zur Applikation *Graph Visualizer*. Zu diesem Projekt existiert ein *GitHub-Repository* (Link vgl. Anhang). Dort sind neben dieser Dokumentation auch alle anderen für das Projekt relevanten Artefakte abgelegt (Quellcode-Dateien, API-Dokumentation, UML-Diagramme).

Bei Fachbegriffen wird in diesem Dokument jeweils die englische Form verwendet (z. B. *Requirement* anstelle von *Anforderung*).

|  |  |
| --- | --- |
| **Verwendete** Technologien und Tools | |
| Programmiersprache: | Java (Java SE 7) | |
| Frameworks: | Java Universal Network/Graph Framework (JUNG) | |
| Entwicklungsumgebung: | Eclipse IDE (Kepler Service Release 2) | |
| Build-Management-Tool: | Apache Maven (Maven-Tools Eclipse) | |
| Version-Control-System: | GIT (GIT-Client Eclipse) | |

Tabelle 1: Technologien und Tools

Das JUNG-Framework stellt viele nützliche Interfaces und Klassen zur Manipulation und Visualisierung von Graphen zur Verfügung.

# Requirements und Use Cases

In diesem Kapitel werden zuerst in kurzer Form die *User Requirements* formuliert (basierend auf der Projektbeschreibung des Betreuers). Dann folgt eine detaillierte Erläuterung der *System Requirements*. Basierend auf den Requirements werden dann die *Use Cases* in einem Diagramm dargestellt.

## User Requirements

Es soll eine Software erstellt werden, welche gerichtete und ungerichtete Graphen erstellen und darstellen kann. Die Software soll es ermöglichen, Graphen aus einer Datei zu laden, zu bearbeiten und in einer Datei zu speichern.

Gleichzeitig soll die Software zur Visualisierung der Traversierung von Graphen dienen. Verschiedene Algorithmen wie z. B. Tiefensuche, Breitensuche, Dijkstra oder Kruskal sollen mit diesem Werkzeug auf einfache Weise visualisierbar werden. Die Visualisierung der Algorithmen soll dabei entweder in einer Animation oder Schritt für Schritt möglich sein. Das Werkzeug soll sich als didaktisches Hilfsmittel für beliebige Graphen-Algorithmen eignen.

## System Requirements

### Funktionale Requirements

* Neuen Graphen erstellen:
  + Gerichtet oder ungerichtet
  + Gewichtet oder ungewichtet (wenn ungerichtet, dann ist das Gewicht bei allen Kanten = 1)
  + Mehrfachkanten und Hyperkanten sind nicht erlaubt
* Graphen und Elemente bearbeiten:
  + Graphen bearbeiten:
    - Name und Beschreibung ändern und anzeigen
    - Verschiebung, Drehung, Scherung
  + Graph-Elemente Bearbeiten
    - Eindeutiger Name (ID) festlegen und anzeigen
    - Knoten:
      * Neuen Knoten erstellen
      * Maximal ein Startknoten festlegen (optional)
      * Maximal ein Endknoten festlegen (optional)
      * Höhe und Breite ändern
      * Position ändern durch Verschiebung
      * Knoten löschen (inzidente Kanten werden automatisch gelöscht)
    - Kanten:
      * Neue Kante zwischen zwei Konten erstellen
      * Gewicht festlegen und anzeigen
      * Kante löschen
* Algorithmus auswählen (Vorberechnung der Visualisierungs-Schritte):
  + Name und Beschreibung des gewählten Algorithmus anzeigen
  + Für DFS, BFS, Dijkstra gilt: Falls kein Startknoten festgelegt wurde, wähle zufällig einen Startknoten.
  + Rekursive Tiefensuche (DFS):
    - Falls ein Endknoten festgelegt wurde und dieser vom Startknoten aus erreichbar ist: Besuche durch *Tiefensuche* alle Knoten bis der Endknoten erreicht wurde und speichere die einzelnen Schritte.
    - Falls *kein* Endknoten festgelegt wurde oder der Endknoten vom Startknoten aus *nicht* erreichbar ist: Besuche durch *Tiefensuche* alle vom Startknoten aus erreichbaren Knoten und speichere die einzelnen Schritte.
  + Breitensuche (BFS):
    - Falls ein Endknoten festgelegt wurde und dieser vom Startknoten aus erreichbar ist: Besuche durch *Breitensuche* alle Knoten bis der Endknoten erreicht wurde und speichere die einzelnen Schritte.
    - Falls *kein* Endknoten festgelegt wurde oder der Endknoten vom Startknoten aus *nicht* erreichbar ist: Besuche durch *Breitensuche* alle vom Startknoten aus erreichbaren Knoten und speichere die einzelnen Schritte.
  + Dijkstra:
    - Falls ein Endknoten festgelegt wurde und dieser vom Startknoten aus erreichbar ist: Suche mit dem *Dijkstra-Algorithmus* den kürzesten Weg vom Start- zum Endknoten und speichere die einzelnen Schritte.
    - Falls *kein* Endknoten festgelegt wurde oder der Endknoten vom Startknoten aus *nicht* erreichbar ist: Suche mit dem *Dijkstra-Algorithmus* den kürzesten Weg vom Startknoten zu allen anderen Knoten und speichere die einzelnen Schritte.
  + Kruskal:
    - Berechne mit dem Kruskal-Algorithmus einen minimal aufspannenden Wald und speichere die einzelnen Schritte.
* Animation anzeigen:
  + Zeitintervall festlegen (in Sekunden)
  + Starten
  + Anhalten
  + abbrechen
* Schrittweise Traversierung:
  + Statusmeldung für jeden Schritt anzeigen
  + Zum Anfang
  + Zum Ende
  + Einen Schritt vorwärts
  + Einen Schritt zurück
* Graphen aus Datei laden
* Graphen in Datei speichern

### Nicht-funktionale Requirements

xml-Format, validierung

Erweiterbarkeit

Usabitlity (shortcuts, save-abfragen

## Use Cases

Nur die wichtigsten Use Cases Diagramm

Neuer Graph extends Graph bearbeiten

Animation extends schritt Traversierung

Laden, speichern include Graph IO

# Systemarchitektur und Design

## Schichtenarchitektur

AA

## MVC-Pattern

## Knoten und Kanten

## Graph-Klassen

## Algorithmen-Klassen

## Step-Klassen

# Implementation und Tests

Der Grossteil des Quellcodes wurde selbst implementiert und ist mit Java-Annotationen entsprechend gekennzeichnet. Alle von Drittpersonen übernommenen Klassen sind mit dem entsprechenden Entwickler der Klasse annotiert. Bei einigen aus dem JUNG-Framework abgeleiteten Klassen wurde teilweise bestehender Code aus der Basisklasse übernommen und angepasst. Solche Klassen sind mit dem eigenen Namen sowie mit dem Namen des Basisklassen-Entwicklers annotiert.

## Coding Conventions

## Algorithmen-Implementation

### Tiefensuche (DFS)

### Breitensuche (BFS)

### Dijkstra

### Kruskal

## Tests

# Benutzer-Dokumentation

API-Doc Hinweis

## Programm

## Graphml-Format

# Abbildungsverzeichnis

[Abbildung 1: Et ut aut isti repuditis qui ium **Fehler! Textmarke nicht definiert.**](#_Toc371572055)

# Tabellenverzeichnis

[Tabelle 1: Technologien und Tools 3](#_Toc390531978)

# Glossar

**Auinweon**

Et ut aut isti repuditis qui ium 7

**Batnwpe**

Et ut aut isti repuditis qui ium 9

**Cowoll**

Et ut aut isti repuditis qui ium 11

# Literaturverzeichnis

**Literatureintrag**

*Autorname, Autorvorname, Buchtitel, Verlag, Ort, Ausgabe, Jahr* 7

**Literatureintrag**

*Autorname, Autorvorname, Buchtitel, Verlag, Ort, Ausgabe, Jahr* 9

**Literatureintrag**

*Autorname, Autorvorname, Buchtitel, Verlag, Ort, Ausgabe, Jahr* 11

# Anhang

Link zum GitHub-Repository : <https://github.com/kofmp1/GraphVisualizer>

Link zur Projekt-Dokumentation (PDF) im GitHub-Repository: <https://github.com/kofmp1/GraphVisualizer/tree/master/ProjectGraphVisualizer/doc>

Link zur API-Dokumentation: <http://htmlpreview.github.io/?https://github.com/kofmp1/GraphVisualizer/blob/master/ProjectGraphVisualizer/doc/apidocs/index.html>

# 