

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Projekt 1

Graph Viewer

Projektdokumentation

Berger Stefan

Menzi Stephan

Gugelmann Christian

# Allgemeines

Dieses Dokument beinhaltet die grundlegenden Anforderungen, welche es im Rahmen des Projektes *Projekt 1* umzusetzen gilt, beschreibt die gemachten Ideen und Überlegungen und deren Umsetzung.

## Projektbeschreibung

Projektbeschreibung gemäss Aufgabenstellung:

*Es soll eine Software erstellt werden, welche Graphen darstellen kann, bzw. es erlaubt, Graphen mit einem graphischen Editor zu spezifizieren. Gleichzeitig soll die Software der Visualisierung der Traversierung von Graphen dienen. Ein Algorithmus, wie etwa derjenige von Dijkstra soll mit diesem Werkzeug so auf einfache Weise visualisierbar werden. Das Werkzeug soll sich als didaktisches Hilfsmittel bzw. als Debugging Tool für beliebige Graphen-Algorithmen eignen.*

## Projektorganisation

|  |  |
| --- | --- |
| Projektteam | Stefan Berger, Stefan Menzi, Christian Gugelmann |
| Projektbetreuung | Peter Schwaab |

# Anforderungen

Das Graph Viewer Tool bietet dem Anwender eine graphische Oberfläche, mit welcher er Graphen erstellen und bearbeiten kann. Erstellte Graphen können abgespeichert und wieder geladen werden. Zusätzlich bietet das Tool die Visualisierung von vordefinierten oder eigenen Graphen Algorithmen anhand der erstellten Graphen an.

## Erstellen und Bearbeiten von Graphen

Graphen können auf einfach Art & Weise erstellt und bearbeitet werden. Dazu werden folgende Funktionalitäten angeboten:

* Hinzufügen/Entfernen von Eckpunkten
* Hinzufügen/Entfernen von Kanten
* Auswahl gerichtete/ungerichtete Graphen
* Benennung der Eckpunkte
* Gewichtung der Kanten
* Verschiebung der Eckpunkte mittels Drag & Drop
* Darstellung der Eckpunkte, Kanten und Labels mittels gewünschter Farbe

## Import/Export von Graphen

Erstellte Graphen können abgespeichert und für eine spätere Verwendung wieder geladen werden. Das Abspeichern erfolgt als XML-File.

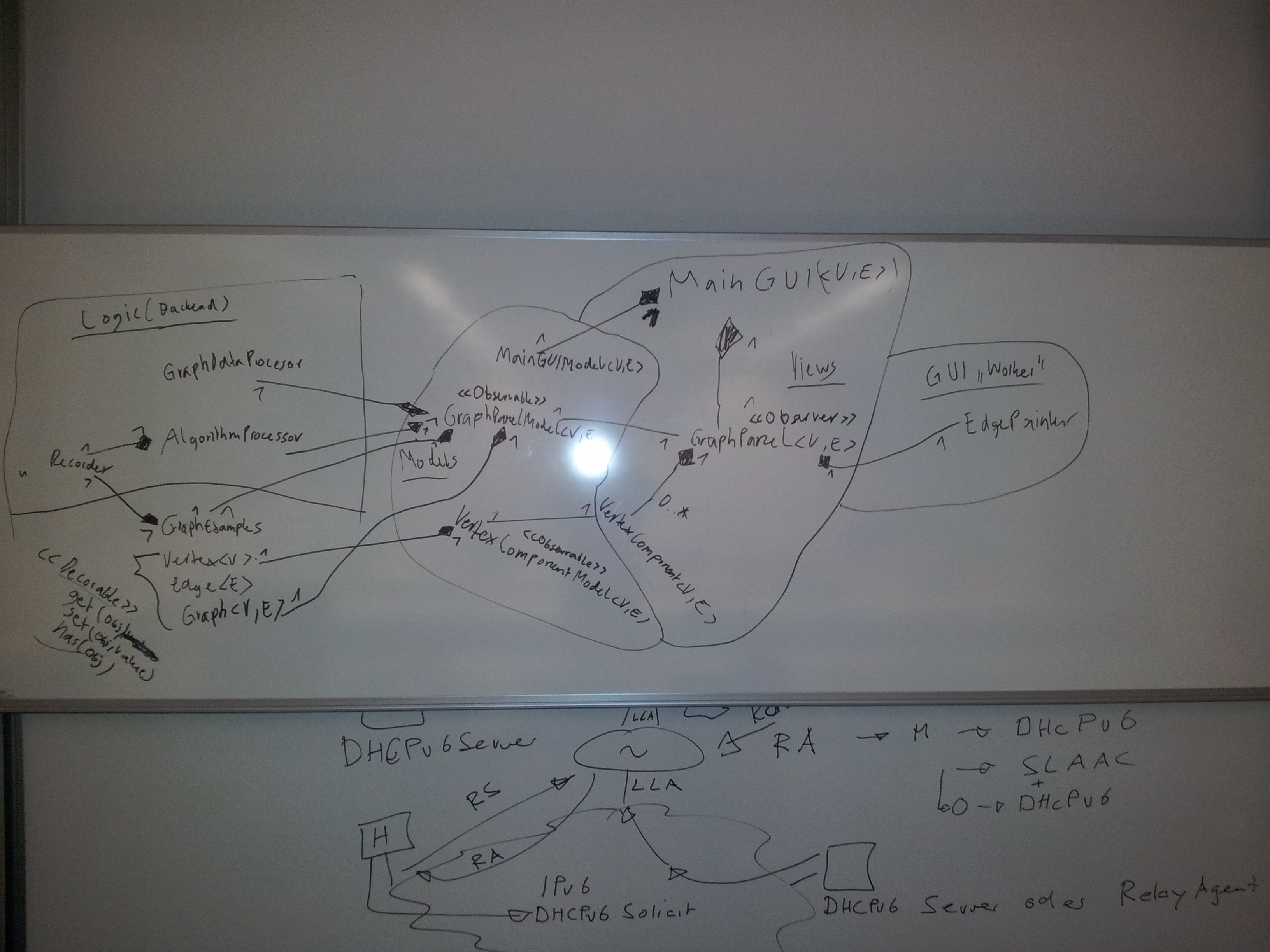
## Visualisierung von Graphen Algorithmen

Vordefinierte (Dijkstra, Kruskal, BFS) oder eigene (Custom Algorithm) Graphen Algorithmen können mittels dem Graph Viewer Tool visualisiert werden. Das Tool bietet die Möglichkeit, die Graphen Algorithmen Step-by-Step zu durchlaufen (vorwärts und rückwärts) und stellt den aktuellen Stand graphisch dar (Highlighting). Dazu gehören:

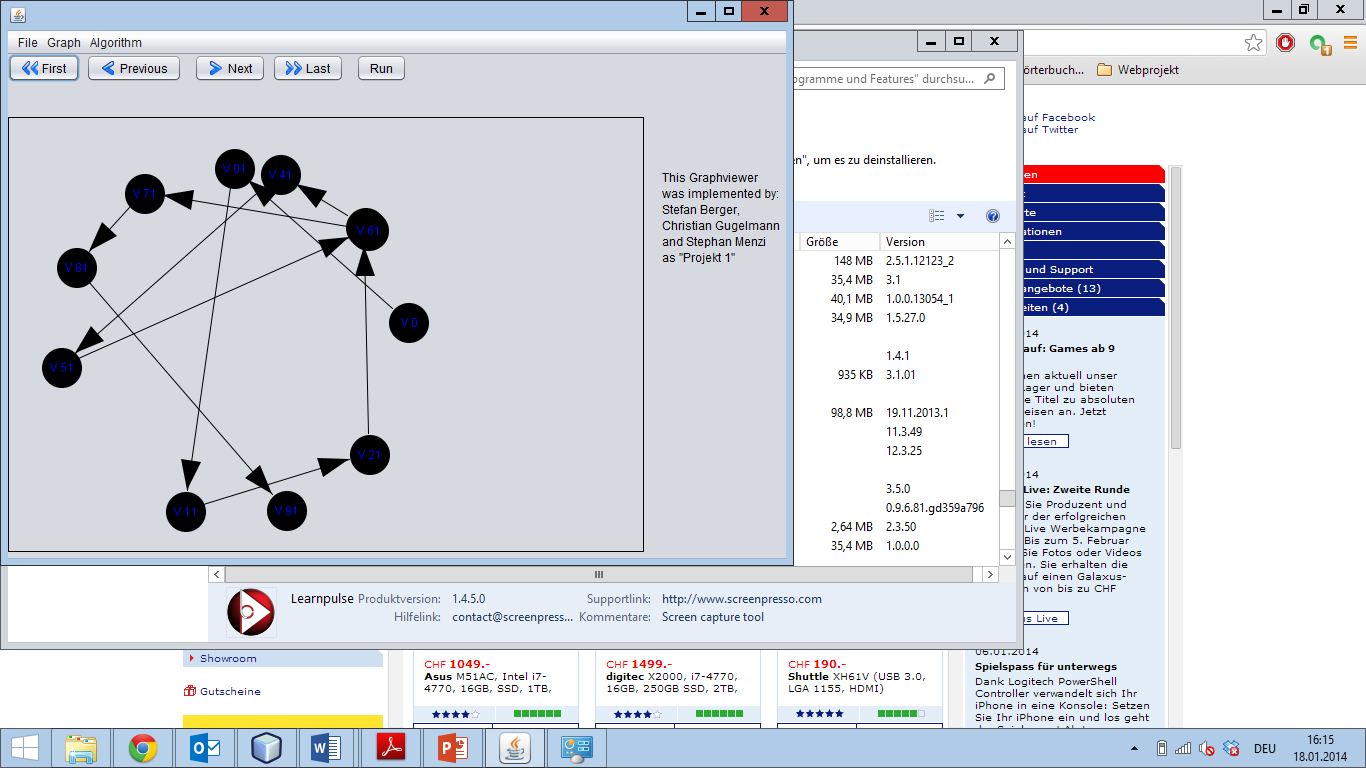
* Aktiver Eckpunkt/aktive Kante
* Besuchter Eckpunkt/besuchte Kante
* ....???!!!

# Umsetzung

## Klassendiagramm



## GUI



Das Haupt-GUI wurde mittels NetBeans-Editor erstellt.

menzi

## Graph, Vertex, Edge

Ein Graph, Vertex und Edge besitzen verschiedene Attribute, welche jeweils in einer separaten Format-Klasse definiert sind.

### Attribute Graph (GraphFormat.java)

* Labels sichtbar/unsichtbar (isLabelVisible)
* directed/undirected (isDirected)
* Gewichtung (isWeighted)
* Aktive Farbe (activeColor)
* Farbe besuchtes Element (visitedColor)
* Farbe unbesuchtes Element (unvisitedColor)

### Attribute Vertex (VertexFormat.java)

* Position Mitte des Punktes (ceterPoint)
* Labe (label)
* Aktiv/inaktiv (active)
* Besucht/unbesucht (visited)
* Distanz (distance)

### Attribute Edge (EdgeFormat.java)

* Aktiv/inaktiv (active)
* Besucht/unbesucht (visited)
* Ausgangspunkt (fromPoint)
* Zielpunkt (toPoint)

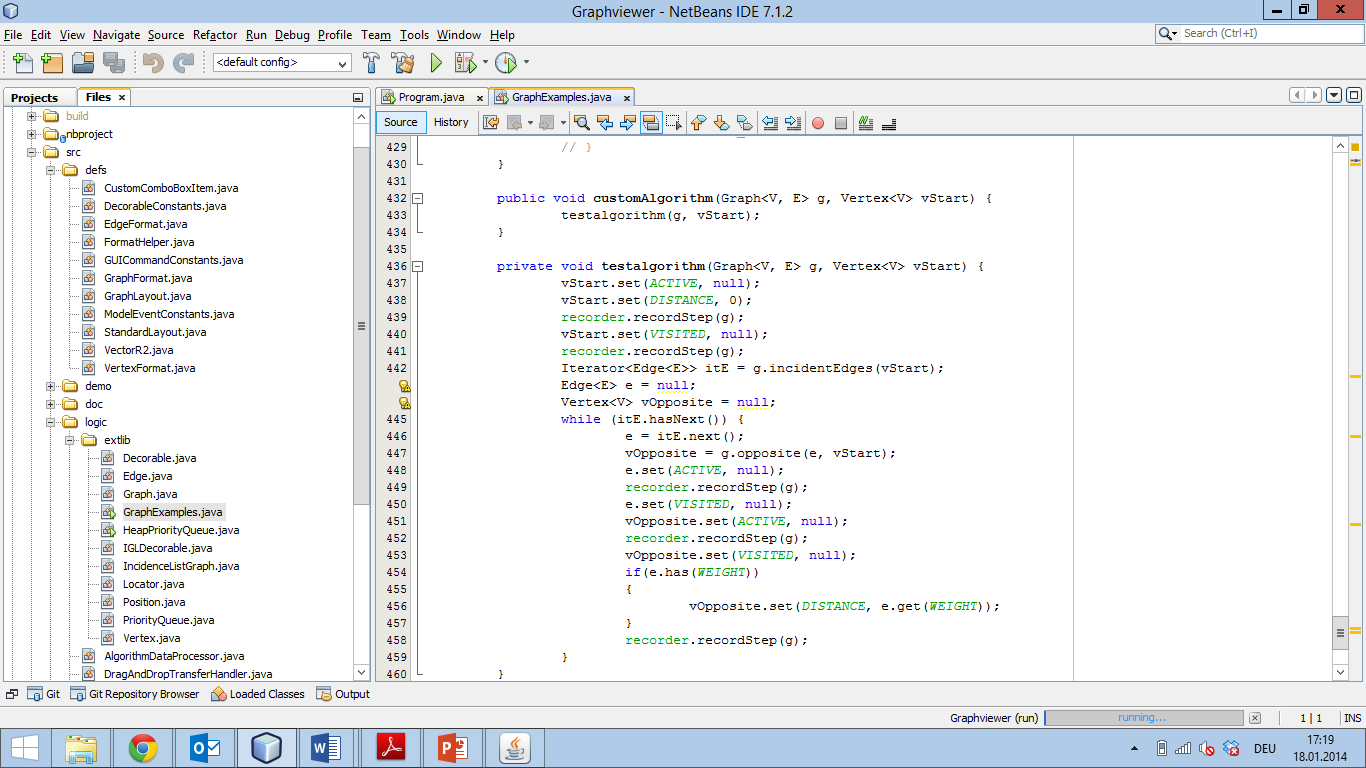
## Graph Editor/Visualisierung Graph

Berger: Grundlegender Aufbau

Menzi: Recorder...

## Custom Algorithm: Attribute zur Visualisierung setzen

Damit ein Graphen Algorithmus vom Graph Viewer Tool visualisiert werden kann, ist es nötig, im entsprechenden Algorithmus (in GraphExamples.java) diverse Elemente korrekt zu setzen, damit diese vom Tool dargestellt werden können. Anhand eines kleinen Testalgorithmus soll dies illustriert werden:



**1**

**2**

|  |  |
| --- | --- |
| **1** | Zu jedem Vertex oder Edge wird das Attribut gesetzt: <Element>.set(ATTRIBUT, WERT) |
| **2** | Der Graph im aktuellen Zustand wird im Recorder gespeichert: recorder.recordStep(g) |

Mögliche Attribute sind:

* ACTIVE
* DISTANCE
* VISITED
* WEIGHT

## Import/Export

Ein Graph wird als XML-File abgespeichert. Die Klasse GraphDataProcessor besitzt dafür die Methode constructStringFromGraph, welcher ein Graph übergeben werden kann und als Ergebnis einen String im XML-Format zurückgibt (mittels eines XML-Parsers (DocumentBuilderFactory) werden die einzelnen Attribute des Graphen als XML-Baum angelegt). Dieser String wird dann mittels FileWriter auf dem Dateisystem am gewünschten Ort abgelegt.

Zum Importieren eines Graphen wird der Methode reconstructGraphFromString ein String im XML-Format übergeben. Als Ergebnis wird ein IncidentListGraph zurückgegeben.

## Spezielles

Pattern...

# Arbeitsaufteilung

Das Erarbeiten des Grundaufbaus des Tools sowie die Definitionen eines Graphen, Vertex und Edges erfolgten im Team. Danach konnten die Arbeiten in drei Hauptkategorien aufgeteilt werden, welche unter den Teammitgliedern verteilt wurden:

* Stefan Berger: Logik & Darstellung Graph
* Stephan Menzi: Realisierung GUI, Recorder
* Christian Gugelmann: Import/Export Graph

Das Projektteam traf sich wöchentlich zur gemeinsamen Besprechung der getätigten Arbeiten und zum Definieren des weiteren Vorgehens. Da grundsätzlich in der BFH implementiert wurde, konnte man sich beim Implementieren gegenseitig unterstützen und allfällige Probleme zusammen lösen.

# Benutzerhandbuch