Implementierung
Graph von Ansicht

Nicolas Boltz
uweaw@student.kit.edu

Jonas Fehrenbach
urdtk@student.kit.edu

Sven Kummetz
kummetz.sven@gmail.com

Jonas Meier
Meierjonas96@web.de

Lucas Steinmann
ucemp@student.kit.edu

Inhaltsverzeichnis

1 Einleitung **3**

2 Änderungen am Entwurf **4**
2.1 Gerneral 5
2.2 Sugiyama 6

3 Implementierte Muss- und Wunschkriterien **7**
3.1 Pflichtkriterien 7
3.2 Wunschkriterien 8

4 Implementierungsplan **9**
4.1 Woche 1 10
4.2 Woche 2 11
4.3 Woche 3 12
4.4 Woche 4 13

5 Unit Tests **14**

1 Einleitung

2 Änderungen am Entwurf

2.1 Gerneral

1. **Aktion:** Rename: GraphMLImporter -> GraphmlImporter
   Grund: Match Conventions (no more than 1 capital letter in abreviations)
2. **Aktion:** Added Generics to Raw-Types: VertexFilter and EdgeFilter
   Grund: Create more type-safety during compile-time.
3. **Aktion:** Moved method getGraphBuilder from IVertexBuilder to IGraphBuilder
   Grund: Better possibility to build hierarchical graphs dynamically
4. **Aktion:** Collabsable Graph Interface hinzugefügt
   Grund: unterscheiden zwischen einem compoundgraph(z.B. FieldAccess) und einem normalen zusammengeklappten subgraphen
5. **Aktion:** JoanaBuilder Abhängigkeiten hinzugefügt
   Grund: Damit jeder Builder weiß von wem er erstellt wurde und sein Produkt bei diesem platzieren kann.
6. **Aktion:** Added AbstractPluginBase
   Grund: Many functions in the concrete Plugins are nearly empty, but have to be overwritten. An AbstractPluginBase reduces identical code.
7. **Aktion:** Removed build() functions from IGraphBuilder, IVertexBuilder, IEdgeBuilder.
   Grund: build() is now only called on IGraphModelBuilder, which then calls recursively the specific build functions of the concrete classes. (Lucas)
8. **Aktion:** Replaced nodeKind (String) field from JoanaVertex with enum. Adapted interface of JoanaVertex accordingly.
   Grund: All reason why using enum is better when possibility are known at compiletime. (Lucas)
9. **Aktion:** Replaced edgeKind (String) field from JoanaEdge with enum. Adapted interface of JoanaEdge accordingly.
   Grund: All reason why using enum is better when possibility are known at compiletime. (Lucas)
10. **Aktion:** SerializedGraph verschoben. Die View wird nun serialized und nicht das model
    Grund: View besitzt mehr Information über den Graphen wie Koordinaten -> wichtig für SvgExporter (Jonas F)
11. **Aktion:** Added class DefaultDirectedEdge and changed DirectedEdge from class to interface. Changed occurences of DirectedEdge to DefaultDirectedEdge in most cases in whole project.
    Grund: There was a need of an interface of DirectedEdge (Jonas M)
12. **Aktion:** Get Color in Edge und Vertex hinzugefügt
    Grund: Die Farbe muss von der Edge/Vertex vorgegeben werden, damit die GUI verschiedene Farben vergeben kann, JOANA Edge und Vertex geben jeweils eine Farbe zurück die von ihrer jeweiligen KIND bestimmt wird.

2.2 Sugiyama

1. **Aktion:** Changed method return type of reverseEdge(SugiyamaEdge edge) in ICycleRemoverGraph from Set<SugiyamaEdge> to void
   Grund: Not really necessary to know which edges have been turned, it can be queried from the edge throug an instance of a SugiyamaGraph (Jonas M)
2. **Aktion:** Added Interface ISugiyamaVertex and let SugiyamaVertex and DummyVertex implement it. Changed every occurence of SugiyamaVertex to ISugiyamaVertex in package sugiyama
   Grund: It's necessary to treat SugiyamaVertex and DummyVertex the same way in a common list (Jonas M)
3. **Aktion:** Moved class Point to from package sugiyama to package edu.kit.student.util
   Grund: For a better overview (Lucas)
4. **Aktion:** SupplementPath does not extends DirectedEdge anymore.
   Grund: ?

3 Implementierte Muss- und Wunschkriterien

3.1 Pflichtkriterien

3.1.1 Allgemein

- Hierarchisches Layout mit dem Sugiyama-Framework
- Ein Callgraph-Layout, welches übersichtlich die Abhängigkeiten der Methoden darstellt
- Ein Methodgraph-Layout, welches die Abhängigkeiten innerhalb einer Methode - mithilfe von vorgegebenen Constraints darstellt
- Kollabieren und Ausklappen von Subgraphen
- Informationsanzeige zu einzelnen Knoten und Kanten
- Statistiken über den Graphen und Subgraphen
- Filter für Knoten- und Kantentypen aus JOANA
- Tabs für geöffnete Graphen
- Akzeptieren von Kommandozeilenargumenten zur Angabe von Graphdatei und Layoutalgorithmus für ein schnelles Starten
- Das Produkt wird unter einer freien Lizenz veröffentlicht
- Die Anzeigesprache der GUI ist englisch. Ein Sprachwechsel soll aber leicht zu implementieren sein.
  - JavaFX bietet bereits eine Möglichkeit verschiedene Sprachen zu verwenden und Oberflächenstrings anhand einer ID zu vergeben.

3.1.2 Input/Output

- Import von generischen Graphen im GraphML-Format
- Export der visualisierten Graphen im SVG-Format

3.1.3 Steuerung

- Navigation mittels Zoom und Verschieben

  - Die im Pflichtenheft definierte Maustastenbelegung kann von Java leider nicht unter jedem Betriebssystem realisiert werden. Im Fall von Windows 8.1 gibt es eine Funktionalität die auf den Klick des Mausrads reagiert. Aus diesem Grund wurde die Tastenbelegung leicht abgeändert:
  - Das Sichtfeld verschiebt man nun durch drücken von STRG und klicken und ziehen der rechten Maustaste.

- Selektieren und Deselektieren von einzelnen oder mehreren Knoten
  - Um das Selektieren und Deselektieren benutzerfreundlicher zu gestalten, wurde die Tastenbelegung an die der gängigen Dateisystemexplorer angepasst:
  - Zum Hinzufügen oder Entfernen eines Knotens aus der Selektion muss beim Klicken STRG mit gedrückt werden, anders wird nur der angeklickte Knoten selektiert.
  - Ist eine Menge von Knoten selektiert, STRG gedrückt und man selektiert durch ziehen der Maus bereits selektierte Knoten, werden diese aus der Selektion entfernt und neue Knoten hinzugefügt.

3.1.4 Plugins

- Schnittstellen für Plugins in den Bereichen Import, Export, Layoutalgorithmen, Filter für Knoten- und Kantentypen und weitere Operationen auf einzelne Knoten und Kanten
- Es gibt ein Pluginmanagement-System, welches externe Plugins laden und verwalten kann.

3.2 Wunschkriterien

Aus den Wunschkriterien die im Pflichtenheft definiert wurden wurden keine Implementiert.

4 Implementierungsplan

Um einen Implementierungsplan aufzustellen wurde das Projekt zuerst anhand der einzelnen Plugins/Pakete und dann noch feiner in einzelne möglichst voneinander unabhängige Aufgaben unterteilt. Der benötigte Arbeitsaufwand zum Bearbeiten der Aufgaben wurden von allen Projektmitgliedern abgeschätzt und deren Mittelwert als geplante Bearbeitungsdauer zur Fertigstellung der Aufgabe angenommen.
Anhand von bestehenden Abhängigkeiten der Aufgaben untereinander wurde folgender Implementierungsplan erstellt:

		Geschätzter Aufwand	Bearbeitung in Woche	Zugewiesen an
objectproperties		4	1	Nicolas
parameter		5	1	Nicolas
graphmodel				
	Interfaces schreiben	2	1	Jonas F.
	Default-Klassen implementieren	8	1	Jonas F.
	Layout-Register schreiben	4	1	Jonas F.
	Serialized-Klassen	2	1	Jonas F.
sugiyama				
	Graph + Impl Interfaces + Vertex/Edge	9	1	Jonas M + Sven
	Phase 1	7	1	Jonas M + Sven
	Phase 2	10	1	Jonas M + Sven
	Phase 3	10	2	Jonas M + Sven
	Phase 4	10	2	Jonas M + Sven
	Phase 5	10	2	Jonas M + Sven
	Constraints implementieren	4	3	
	LayoutAlgorithm	2	3	
joana				
	Callgraph	5	2	Lucas
	Methodengraph	5	2	Lucas
	FieldAccess	3	2	Lucas
	GraphModel	4	2	Lucas
	CallgraphLayout/Register/Option	13	3	
	MethodengraphLayout/Register/Option	13	3	
	GraphBuilder-Klassen	12	3	
	JoanaPlugin	6	3	
	Joana Vertex/Edge	4	2	Nicolas
	Workspace	5	3	
	JoanaGraph	2	2	Nicolas
GUI				
	GraphFactory	5	3	
	GraphView(Navigation und Zoom)	10	3	
	StrukturView	4	4	
	InformationView	4	4	
	Generelle GUI	11	1	Nicolas
	Laden der Plugins	6	2	Nicolas + Lucas
	Import und Export anstoßen + Dialoge	5	2	Nicolas
	ParameterDialogCreator	6	3	
Plugin				
	Manager	11	1	Lucas
	Interfaces schreiben	3	1	Lucas
	Filter	7	3	
	Workspace-/EntrypointOption	4	3	
Import		11	2	Jonas F.
Export		7	4	

Abbildung 4.1: Erster Implementierungsplan

4.1 Woche 1

In der ersten Woche wurde der zuerst erstellte Implementierungsplan komplett überarbeitet. Die Aufgaben wurden in der Abarbeitungsreihenfolge umsortiert, verfeinert und für die ersten zwei Wochen einzelnen Personen zugewiesen. Zusätzlich wurde eine Tabelle erstellt, die den bereits existierenden Plan, durch eine Fortschrittsspalte, in der die Entwickler ihren Prozentualen Fortschritt der ihnen zugeteilten Aufgaben eintragen, und eine visuelle Repräsentation des Soll- und Ist-Zustands des gesamten Projekt erweitert.

Zusätzlich zur Änderung des Plans wurde eine grobe Projektstruktur in Gradle aufgebaut und die bereits im Entwurf gemachten Interfaces in das Projekt überführt. Um die Gradle Projektkonfiguration zu testen wurde eine grobe GUI implementiert und die Entwicklung der allgemeinen genutzten Klassen aus dem Paket ”graphmodel”begonnen.

Paket	Aufgabe	Aufwand(h)	Woche	Zugewiesen an	Fortschritt in %	Datum der Fertigstellung
Allgemein	Entscheidung Namespace	1	1	Alle	100	
Allgemein	Erstellen der Gradle-Konfiguration	6	1-3	Lucas	75	
objectproperties	Implementieren	4	1	Nicolas	100	23.06.16
parameter	Implementieren	5	1	Nicolas	100	23.06.16
graphmodel	Interfaces schreiben	2	1	Jonas F + Lucas	50	
graphmodel	Default-Klassen implementieren	8	1	Jonas F (+ Sven)	60	
graphmodel	Layout-Register schreiben	4	1	Jonas F	20	
graphmodel	Serialized-Klassen	2	1	Jonas F (+ Sven)	70	
sugiyama	Graph + Impl Interfaces + Vertex/Edge	9	1	Jonas M + Sven	10	
sugiyama	Mokups für einzelne Phasen	8	1	Jonas M + Sven	0	
sugiyama	Constraints implementieren	4	1	Jonas M + Sven	0	
sugiyama	LayoutAlgorithm	2	1	Jonas M + Sven	0	
sugiyama	CycleRemover (Phase 1) + Interface	7	1	Jonas M + Sven	0	
gui	GraphFactory	5	1	Nicolas	75	
gui	ParameterDialogCreator	6	1	Nicolas	100	23.06.16
plugin	Manager	11	1	Lucas	50	
plugin	Interfaces schreiben	3	1	Lucas	0	
gui	Generelle GUI(Incl. Basis-GraphView)	11	1-3	Nicolas	30	

Abbildung 4.2: Überarbeiteter Plan für Woche 1 mit Fortschritt

4.1.1 Verzögerungen

Wie man in 4.3 sehen kann hing das Projekt nach der ersten Woche bereits ca. 35 Stunden hinter dem geplanten Zustand hinterher. Dies lag neben der Tatsache das die erste Hälfte der Woche, auf Grund von Erschöpfung aus der Entwurfsphase, weniger gearbeitet wurde, auch daran, dass das Team das für die Implementierung des Sugiyama-Layoutalgorithmus eingeteilt wurde relativ schnell auf konzeptionelle Probleme mit dem im Entwurf definierten Aufbau des SugiyamaGraphen und dessen Kanten/Knoten stieß(siehe 2.2).

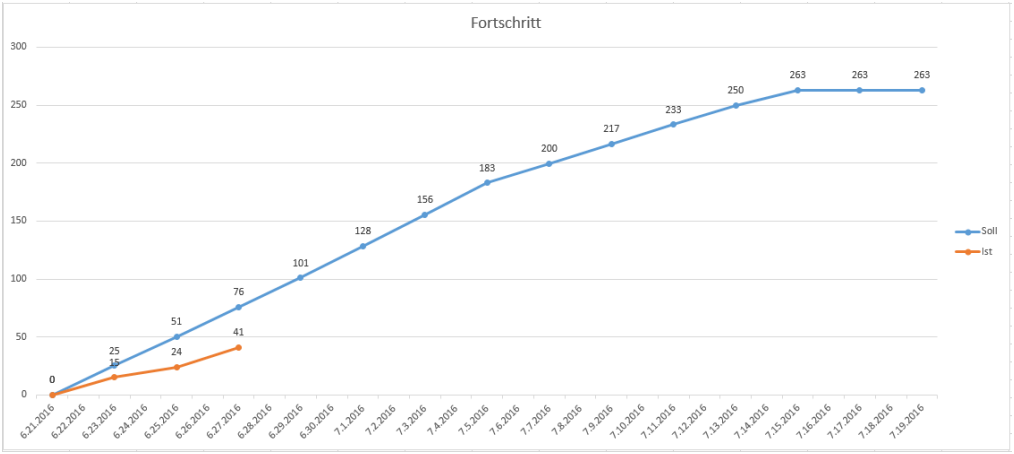

Abbildung 4.3: Kanteninformation Anzeigen

4.2 Woche 2

In Woche 2 wurde vorrangig daran gearbeitet einen Graphen einzulesen, mit den im Sugiyama implementierten Klassen oder Mockups zu layouten und in der Anzeige darzustellen. Dazu wurde besonders auf die zeitnahe Fertigstellung des Imports und das funktionsfähig machen des JOANA-Graph-Plugins mit allen dazugehörigen Klassen geachtet. Zudem wurde auch in den Phasen des Sugiyama-Algorithmus große Fortschritte gemacht und manche sogar abgeschlossen. Dadurch entstand der, wie in 4.5 zu sehen ist, sprunghafte Anstieg des Fortschritts der Fast bis an den Soll-Wert heran reicht.

Paket	Aufgabe	Aufwand(h)	Woche	Zugewiesen an	Fortschritt in %	Datum der Fertigstellung
Allgemein	Erstellen der Gradle-Konfiguration	6	1-3	Lucas	75	
graphmodel	Default-Klassen implementieren	8	1	Jonas F (+ Sven, Nicolas)	90	
graphmodel	Serialized-Klassen	2	1	Jonas F (+ Sven)	70	
sugiyama	Graph + Impl Interfaces + Vertex/Edge	9	1	Jonas M + Sven	70	
sugiyama	Mokups für einzelne Phasen	8	1	Jonas M + Sven	90	
sugiyama	CycleRemover (Phase 1) + Interface	10	1	Jonas M + Sven	70	
gui	GraphFactory	5	1	Nicolas	80	
plugin	Manager	11	1-3	Lucas	50	
gui	Generelle GUI(Incl. Basis-GraphView)	11	1-3	Nicolas	70	
sugiyama	LayerAssigner (Phase 2) + Interface	10	2	Jonas M + Sven	100	04.07.16
sugiyama	CrossMinimizer (Phase 3) + Interface	10	2	Jonas M + Sven	100	04.07.16
joana	Callgraph	5	2	Lucas	70	
joana	Methodengraph	5	2	Lucas	70	
joana	FieldAccess	3	2	Lucas	70	
joana	GraphModel	4	2	Lucas	100	
joana	Joana Vertex/Edge	4	2	Nicolas	100	
joana	Workspace	5	2	Nicolas(+ Lucas)	75	
joana	JoanaGraph	2	2	Nicolas	80	
joana	GraphBuilder-Klassen	12	2	Nicolas + Lucas (+ Jonas F)	50	
gui	Import und Export anstoßen + Dialoge	5	2	Nicolas + Jonas F	100	
gui	Laden der Plugins	6	2	Lucas	100	
import	Implementieren	11	2	Jonas F(+ Nicolas)	100	
sugiyama	VertexPositioner (Phase 4) + Interface	10	3	Jonas M + Sven	40	
sugiyama	EdgeDrawer (Phase 5) + Interface	10	3	Jonas M + Sven	10	

Abbildung 4.4: Implementierungsplan für Woche 1 und 2 mit Fortschritt

4.2.1 Verzögerungen

Aufgrund der im Entwurf definierten Generics im GraphModel entstanden häufig sehr unschöne Stellen im Code und Inkompatibilitäten. Diese zu reparieren oder funktionsfähig zu machen war häufig Grund für mehr oder weniger unschöne Workarounds die Teilweise viel Arbeit in Anspruch nahmen. Deshalb wurde im Laufe der folgenden Woche ein großes Refactoring vorgenommen welches die Generic-Struktur im GraphModel stark vereinfachte.

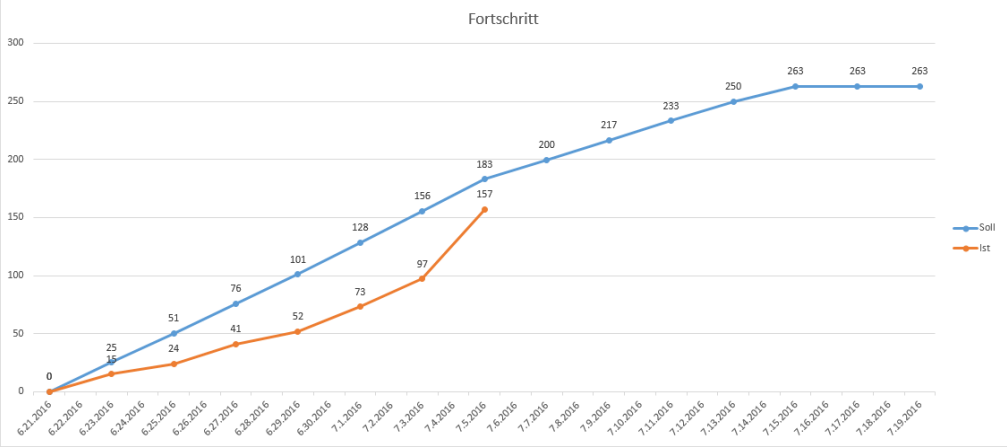

Abbildung 4.5: Kanteninformation Anzeigen

4.3 Woche 3

Paket	Aufgabe	Aufwand(h)	Woche	Zugewiesen an	Fortschritt in %	Datum der Fertigstellung
Allgemein	Erstellen der Gradle-Konfiguration	6	1-3	Lucas	75	
graphmodel	Default-Klassen implementieren	8	1	Jonas F (+ Sven, Nicolas)	95	
sugiyama	Graph + Impl Interfaces + Vertex/Edge	9	1	Jonas M + Sven	85	
gui	GraphFactory	5	1	Nicolas	90	
plugin	Manager	11	1-3	Lucas	70	
gui	Generelle GUI(Incl. Basis-GraphView)	11	1-3	Nicolas	80	
joana	Methodengraph	5	2	Lucas	70	
joana	FieldAccess	3	2	Lucas	70	
joana	Workspace	5	2	Nicolas(+ Lucas)	75	
joana	JoanaGraph	2	2	Nicolas	90	
sugiyama	VertexPositioner (Phase 4) + Interface	10	3	Jonas M + Sven	50	
sugiyama	EdgeDrawer (Phase 5) + Interface	10	3	Jonas M + Sven	10	
joana	JoanaPlugin	6	3	Lucas	75	
joana	CallgraphLayout/Register/Option	7	3	Nicolas + Jonas F	80	
joana	MethodengraphLayout/Register/Option	7	3	Nicolas + Jonas F	40	
gui	GraphView(Navigation, Selection und Zoom)	10	3	Nicolas + Jonas F	90	
plugin	Filter	7	3	Lucas	0	
plugin	Workspace-/EntrypointOption	4	3	Lucas	100	
gui	StrukturView	4	4	Nicolas	90	
gui	InformationView	4	4	Nicolas	70	
gui	Kommandozeilenparameter akzeptieren	4	4	Jonas F	0	
export	Implementieren	7	4	Jonas F	70	

Abbildung 4.6: Kanteninformation Anzeigen

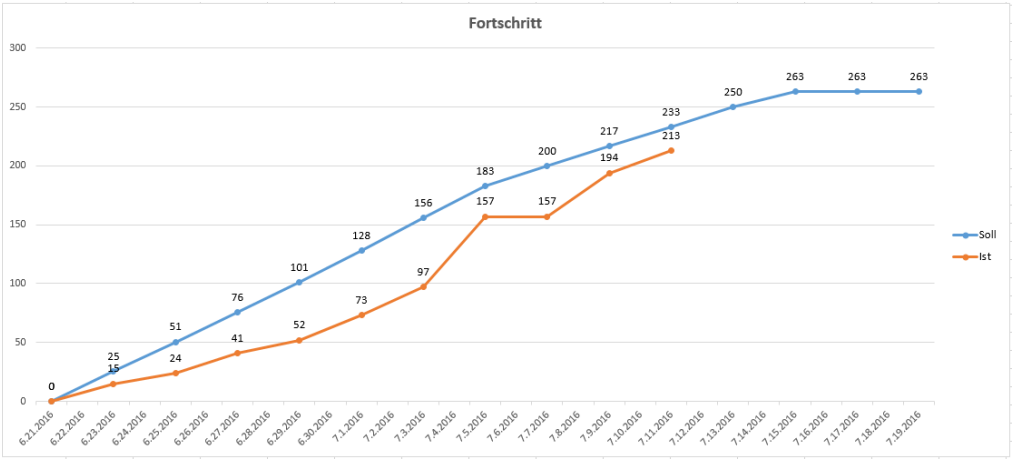

Abbildung 4.7: Kanteninformation Anzeigen

4.3.1 Verzögerungen

4.4 Woche 4

4.4.1 Verzögerungen

5 Unit Tests