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

2 Änderungen am Entwurf

2.0.1 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 compile-time. (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 compile-time. (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)

2.0.2 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.

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
- Selektieren und Deselektieren von einzelnen oder mehreren Knoten

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

3.2.1 Allgemein

- Automatisiertes Subgraph finden mittels Graph Pattern Matching
- Layout Constraints, die vom Nutzer angepasst werden
- Fortschrittsbalken bei der Berechnung des Layouts des Graphen
- Eine Übersicht des angezeigten Graphen
- Algorithmus zur Erreichbarkeit eines Knoten
- Die Darstellung von Kanten kann geändert werden
- Reload-Funktion

3.2.2 Steuerung

- Tastaturkürzel (evtl. benutzerdefiniert)

3.2.3 Input/Output

- weitere Exportfunktionen für das JPG- und das GraphML-Format (mit Koordinaten der Knoten und Kanten des aktuellen Layouts)

4 Implementierungsplan

4.1 Verzögerungen

5 Unit Tests