Projekt AGB-10

Fremdprojektanalyse

Verantwortliche: Marcel Pfütze, Stefanie Enge
Gruppe: SWP10-9

17. Mai 2010

Inhaltsverzeichnis

1 Allgemeines

SONIVIS:Tool ist eine Anwendung zur Netzwerkanalyse und zum Netzwerk Mining. Es ist eine freie Open-Source Software unter der GNU General Public License.

2 Produktübersicht

Die Fremdprojekt hat in erster Linie das Ziel, zu verstehen, wie man die Prefuse Library in größere Projekte eingliedert. Aus diesem Grund handelt es sich hier nicht um eine vollständige Abhandlung der Funktionalität und Struktur von SONIVIS:Tool. Vielmehr werden die Teile näher betrachtet, die direkt mit der Prefuse Library zu tun haben bzw. die mit dem Problem des Comorbidity Graph Browsers zu tun haben.
Am Wichtigsten ist daher die Analyse der folgenden Teilprojekte: de.sonivis.tool.core und de.sonivis.tool.view. Deren Aufgaben sind einerseits die Darstellung der Graphen und andererseits die Verwaltung der Datenstrukturen.
In neueren Versionen von SONIVIS:Tool findet die Prefuse Library keine Verwendung mehr, da es Kompatibilitätsprobleme mit dem von SONIVIS:Tool verwandten und neu eingeführten Standard Widget Library gab. Die Klassen, die Prefuse verwenden, sind im aktuellen Quellcode allerdings noch enthalten und können auch analysiert werden.

3 Grundsätzliche Struktur und Entwurfsprinzipien für das Gesamtsystem

3.1 Die Prefuse Library

Die Prefuse Library ist ein in Java geschriebenes Toolkit zur Visualisierung und Verwaltung von Informationen, insbesondere von Graphen. Es bietet Datenstrukturen für Graphen, Bäume und Tabellen sowie Techniken zur Visualisierung und Animation. Sie nutzt dabei das Swing Interface, das seit Version 1.2 Bestandteil der Java Runtime ist.

3.2 Das Standard Widget Toolkit

SONIVIS:Tool nutzt im Gegensatz zu Prefuse die SWT Library zur Darstellung grafischer Oberflächen. SWT ist im Gegensatz zu Swing nicht fester Bestandteil der Java Runtime. Ein weiterer Unterschied ist, dass SWT die nativen grafischen Elemente des Betriebssystems nutzt und sich daher dem Look and Feel des jeweiligen Betriebssystems anpasst.
Die prominentesteste Anwendung, die SWT nutzt, ist die IDE Eclipse.

3.3 Die Projekte von SONIVIS:Tool

Der Quellcode SONIVIS:Tool ist in mehrere Projekte aufgeteilt. Diese enthalten wiederum mehrere Pakete. Für diese Analyse ist vor allem die Nutzung der Prefuse Library interessant, weshalb wir uns den beiden Projekten von SONIVIS:Tool zuwenden, die am meisten mit Prefuse in Berührung kommen.
Das Beispiel der Erstellung eines Visualization Objekts zeigt, dass diese beiden Projekte sehr eng miteinander verwoben sind:
Die Klasse VisualizationView aus dem Paket de.sonivis.view.views ruft bei der Klasse GraphVisualizationFactory die Methode getVisualizationInstance() auf, um ein Objekt der Klasse AbstractVisualization zu bekommen. Die Klasse GraphVisualizationFactory ruft wiederum die Klasse ExtensionPointManager auf, welche die Klasse ModelManager nutzt, um auf den aktuell gespeicherten Graph zuzugreifen. Die drei zuletzt genannten Klassen sind alle Teil des de.sonivis.tool.core Projekts.
Die Klasse GraphView aus dem de.sonivis.tool.view.prefuse Paket ist von AbstractVisualization abgeleitet und das folgende Diagramm lässt sich somit auch auf frühere Versionen übertragen, in denen Prefuse noch zur Darstellung der Graphen verwendet wurde.

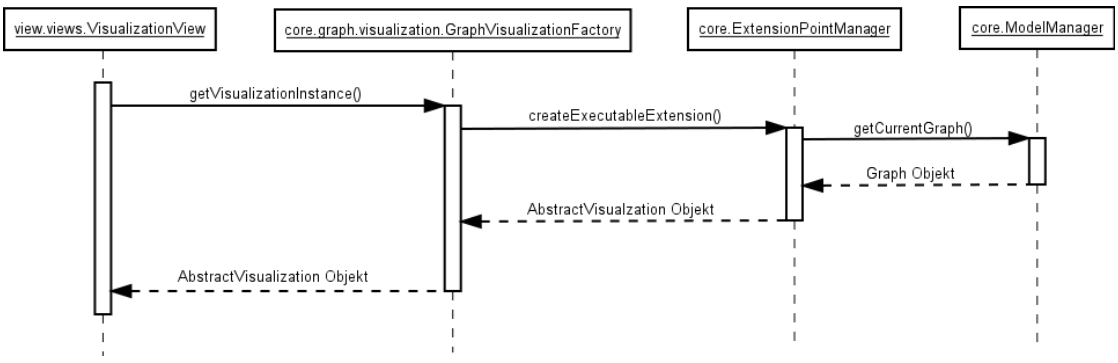

Abb. 1: Sequenzdiagramm zeigt die Erstellung eines Visualization Objekts (de.sonivis.tool wurde im Pfad der Klassen weggelassen).

Der Grund für den regen Informationsfluss zwischen diesen beiden Projekten ist, dass das Projekt de.sonivis.tool.core, wie der Name bereits sagt, Kernfunktionalität zur Verfügung stellt und zusammenfasst. Diese wird von mehreren Teilen des SONIVIS:Tools benötigt. Die Pakete innerhalb von de.sonivis.tool.core haben daher nur wenige Beziehungen untereinander. Dafür wird viel Funktionalität nach aussen zur Verfügung gestellt.
Dies widerspricht der weitläufigen Meinung, dass Pakete Funktionalität möglichst kapseln sollen und dass nach außen nur wenig Informationsfluss bestehen darf. In diesem Fall haben sich die Entwickler von SONIVIS:Tool allerdings dafür entschieden die Übersichtlichkeit zu erhöhen, indem Grundfunktionalität zusammengefasst wird.

3.3.1 Das Projekt de.sonivis.tool.core

Dieses Projekt enthält in erster Linie die Datenstrukturen von SONIVIS:Tool. Zudem werden noch einige Funktionen bereit gestellt, die allerdings mit der Prefuse Library nur peripher agieren. Wichtig für unser Projekt ist, zu erkennen wie Prefuse Verwendung findet. Dazu gehört natürlich auch, wie die Datenstrukturen in SONIVIS:Tool realisiert wurden.

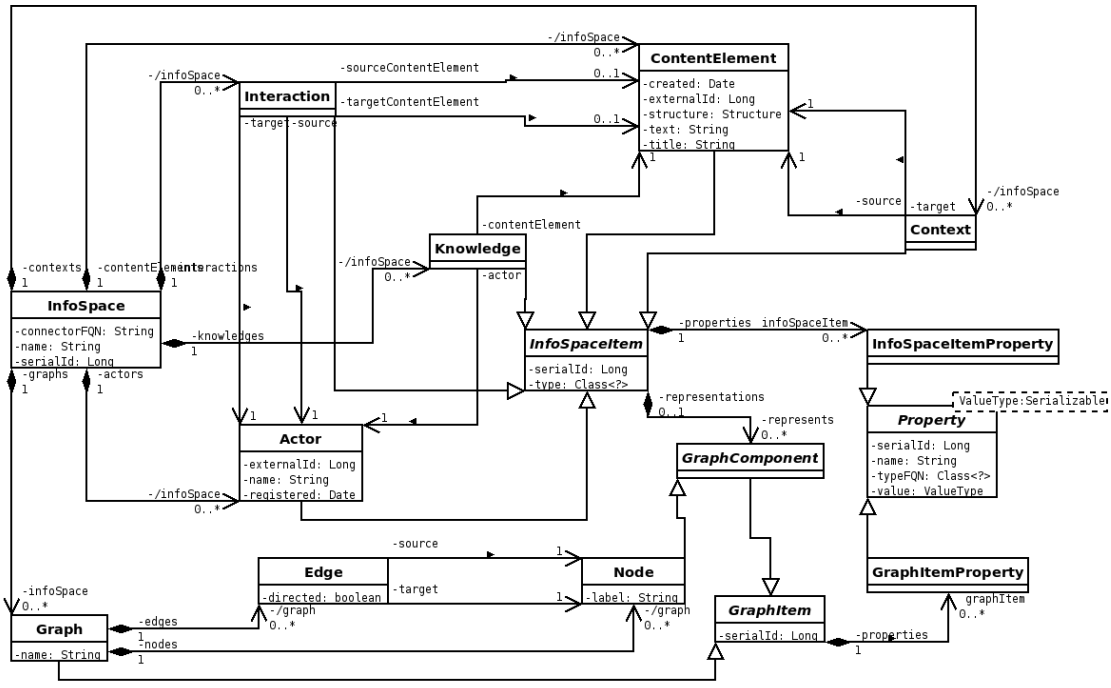

Abb. 2: Klassendiagramm des Datenmodells

Dieses Klassendiagramm stammt aus einer älteren Version von SONIVIS:Tool, in der die Prefuse Library noch Verwendung fand, und ist daher für uns von größerem Interesse als ein neueres.
Man kann gut erkennen, dass die zentrale Klasse InfoSpaceItem ist, und von ihr, teilweise über Umwege, alle Klassen abgeleitet werden.

3.3.2 Das Projekt de.sonivis.tool.view

Dieses Projekt sorgt für die Darstellung der Graphen und nutzt dabei die Prefuse Library. Hierbei bildet das Paket sonivis.tool.view.prefuse die Schnittstelle zwischen SONIVIS:Tool und Prefuse.

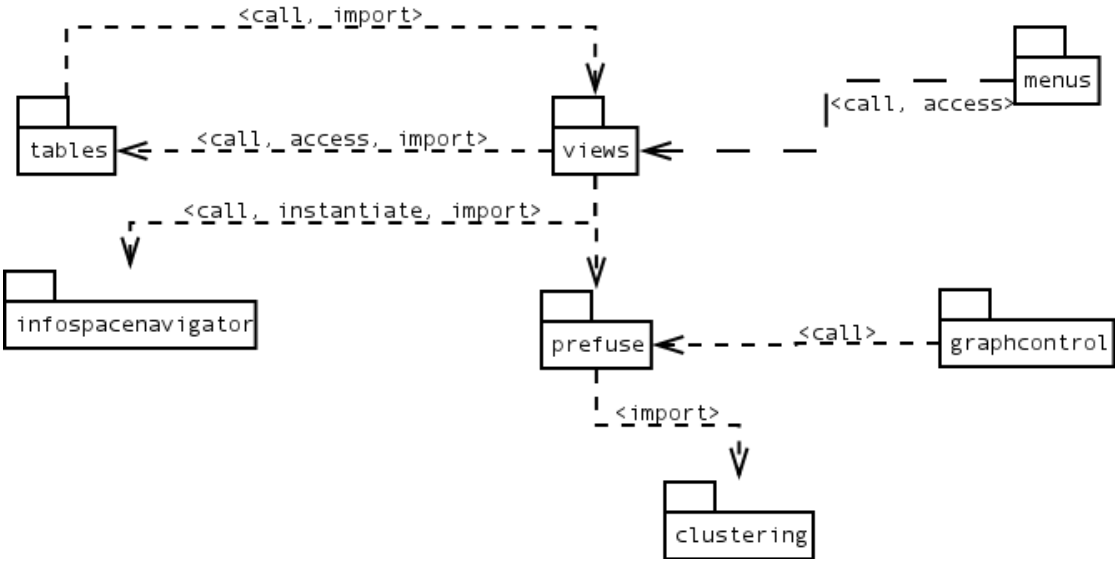

Abb. 3: Paketdiagramm des Projekts sonivis.tool.view

4 Grundsätzliche Struktur und Entwurfsprinzipien der einzelnen Pakete

4.1 Das Paket de.sonivis.tool.view.prefuse

4.1.1 Die Klasse SonivisPrefuseMapping

Die Klasse enthält Hashmaps, die die von Sonivis verwendeten Datenstrukturen für Knoten und Kanten auf die entsprechenden Pendants von Prefuse abbilden. Dem Konstruktor der Klasse werden dabei keine Daten übergeben. Stattdessen werden die verwendeten Hashmaps über die zugehörigen addNode bzw. addEdge Methoden aufgebaut.
Zusätzliche Funktionalität ist die Überprüfung, ob bestimmte Knoten oder Kanten bereits in der Hashmap enthalten sind.

4.1.2 Die Klasse GraphView

Die Klasse Graphview stellt den Graphen in einem Panel dar und bildet somit den Kern dieses Pakets. Es ist von JPanel abgeleitet und nutzt damit das Swing Interface. Zudem implementiert es die Interfaces IcontrolPanelChangedListener aus dem gleichen Paket, Ilistener aus dem sonivis.tool.core.eventhandling Paket und PropertyChangeListener aus dem java.beans Paket.

wichtige Methoden:

GraphView(Graph g, SonivisPrefuseMapping mapping) Der einzige Konstruktor der Klasse. Übergeben wird ein Graph Objekt aus der Prefuse Library und ein SonivisPrefuseMapping Objekt.
Die wichtigsten Funktionen sind das Erstellen und Initialisieren des Visualization Objekts aus der Prefuse Library und das Anmelden bei den implementierten Listenern.

void redraw() Dies ist eine private Methode, die alle Elemente des Visualization Objekts neu zeichnet. Die Methode ist überladen, so dass man auch einzelne Elemente neu zeichnen kann.

4.1.3 Die Klasse PrefuseGraphVisualization

Hierbei handelt es sich um eine Wrapperklasse für die Visualization Klasse aus der Prefuse Library. Aus dem Paket sonivis.tool.core.graph.visualization wird von der AbstractVisualization Klasse geerbt. Sie verwendet intern ein SonivisPrefuseMapping Objekt, um mit entsprechenden Methoden die SONIVIS:Tool bzw. Prefuse Datenstrukturen für Knoten und Kanten zurückzugeben.

wichtige Methoden:

PrefuseGraphVisualization() Der einzige Konstruktor der Klasse. Er erzeugt ein Objekt der Klasse Graph aus der Prefuse Library, das 100 Knoten, aber keine Kanten enthält. Außerdem werden dem Knotenschema eine Spalte für das Kantengewicht und eine Spalte für die Namen der Knoten hinzugefügt, wobei die Standardwerte 1,0 und ein leeres Objekt der Klasse String sind.

4.2 Das Paket de.sonivis.tool.core

4.2.1 Die Klasse Edge

Edge ist eine Klasse, die dazu verwendet wird, eine Verbindung zwischen zwei Knoten darzustellen (eine Kante). Dazu dienen die Attribute **source** und **target** vom Typ Node. Ersteres stellt dabei den Ausgangsknoten und **target** den Endknoten dar. Ob die Verbindung zwischen den beiden Knoten gerichtet ist, wird mit dem Attribut **directed** festgelegt.
Um die Attributwerte zu verändern, existieren getter- und setter-Methoden.
In der Klasse Edge gibt es viele verschiedene Konstruktoren, bei denen sich jeweils die übergebenen Attribute entsprechend ändern. In allen wird jedoch das Attribut **graph** vom Typ Graph übergeben, um zu zeigen, zu welchem Graphen die Kante gehört. Dies ist also ähnlich gelöst wie in der Prefuse Library. Allerdings ist im Gegensatz zu der Implementierung in Edge kein Interface.

wichtige Methoden:

int hashCode() Die Methode überschreibt die von Object geerbte Methode. Die eindeutigen Integerwerte werden zur Abbildung auf Prefuse in der SonivisPrefuseMapping-Klasse benötigt.

4.2.2 Die Klasse Node

Die Klasse Node repräsentiert einen Knoten in einem Graphen des Datenmodells von SONIVIS:Tool. Als Attribut besitzt die Klasse nur einen String **Label**. Dieses dient zur einzigartigen Identifizierung eines Knotens, wenn eine neue Instanz erstellt wird. Im Gegensatz zur Implementierung in Prefuse ist auch die Klasse Node kein Interface in SONIVIS:Tool.

wichtige Methoden:

Node(final Graph graph, final String label) Hier wird nur der Graph und ein Label übergeben. Wenn das Label nicht Null und nicht leer ist, wird eine Instanz des Knotens erzeugt.

int hashCode() Wie auch in der Klasse Edge wird die geerbte hashCode() Methode überschrieben. Der Wert wird in der Methode SonivisPrefuseMapping verwandt.

4.2.3 Die Klasse AbstractVisualization

Die Klasse AbstractVisualization ist abstrakt. Daher können hiervon keine eigenen Objekte gebildet werden. In dieser Klasse befinden sich setter-Methoden für die Labels der Edges, Nodes und Graphen. Außerdem gibt es Setter für die Sichtbarkeit der Kanten, Knoten und Graphen. Auch werden die Farben für die Kanten festgelegt. Zusätzlich lässt sich über eine Methode bestimmen, ob eine Kante gerichtet ist oder nicht.

4.2.4 Die Klasse ModelManager

Der ModelManager wird dazu verwendet, um einzelne Instanzen der Anwendung zu verwalten. So zum Beispiel das Graph Objekt, das aktuell angezeigt wird, oder den aktuellen InfoSpace. Hier wird ein Singleton-Konstruktor verwendet, um sicherzustellen, dass nur genau eine Instanz von dieser Klasse erstellt werden kann. Dazu gibt es eine getMethode, um die Instanz an äußere Klassen ausgeben zu können.