Entwurfskonzept
SONIVISTool

Allgemeines

Das SONIVISTool ist eine javabasierende, Open-Source Applikation, das als Plug-In, auf dem Eclipse-Framework basiert.

Produktübersicht

SONIVIS:Tool ermöglicht Wikis, Webblogs und Soziale Netzwerke darzustellen und zu analysieren. Sonivis bietet die Möglichkeit den Graphen zu untersuchen womit verschiedene Entwicklungsstände der Netzwerke erkannt werden sollen.
Der Schwerpunkt liegt bei Wiki – Netzwerken, die nach verschiedenen Aspekten wie z.b. Zusammenhänge zwischen Informationen oder Personen untersucht werden können.
Die Benutzeroberfläche von Sonivis besteht im wesentlichen aus 3 Komponenten:: Analysis View, Manipulation View und Statistic View. Es gibt noch verschiedene andere Analyse Views für Wikis auf die jetzt nicht genauer eingegangen wird.
Sonivis bietet verschiedene Schnittstellen zum Datenimport. Momentan können nur Daten aus mediawiki Formaten importiert werden. Weiter Import Möglichkeiten sind in Arbeit.
Allerdings bietet Sonivis eine Schnittstellen zur Implementierung einer Importmöglichkeiten für eigene Formate.

Grundsätzliche Struktur- und Entwurfsprinzipien des Systems

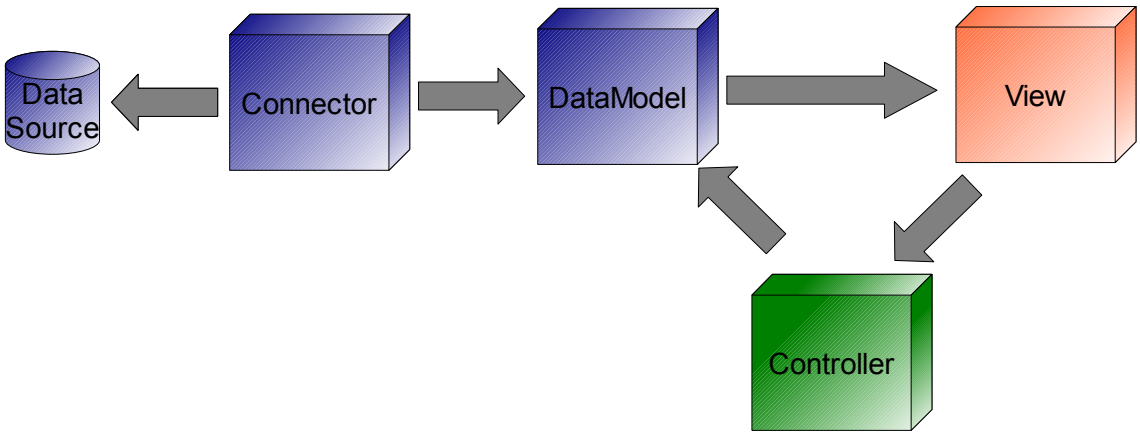
Grundkonzept

Das Grundkonzept, auf dem die Applikation basiert, ist das MVC-Pattern, dass heißt Datenlogik, GUI-Logik und Kontrolllogik werden gekapselt. Das weiterführende Modell des MVC-Patterns ist das IVR-Modell, wobei die Datenlogik weiter unterteilt wird, in Rohdaten, Speicherung der Daten in einem Datenmodell und der Speicherung von Zusatzinformationen, die für die Visualisierung auf der GUI eine Rolle spielen, Visual-Model.

Aufbau des Systems

Aufgrund des IVR-Modell ist im allgemeinen die Kapselung der Schichten in der Anwendung realisiert wurden.
Das Herzstück von Sonivis ist das Datenmodell auf dem die Visualisierung der Daten als Graph basiert. Dieses Datenmodell wurde von Sonivis entwickelt, dass es ermöglicht beliebige Daten so generisch wie möglich darstellen zu können unabhängig von der äußeren Form der Rohdaten. Diseses Datenmodell umfasst sowohl die Daten der "Realität" als auch die Präsentationsdaten, dass heißt die abstrakten Datenstrukturen Graph, Edge und Kante. Nach dem IVR-Modell sollten Quelldaten und Daten, die Informationen der Visualisierung speichern, getrennt voneinander behandelt werden, aber aufgrund der starken Korrespondenz zwischen Daten der "Realität" und Daten der Visualisierung, werden diese in einem Datenmodell realisiert. Durch themenspezifische Connector werden die Rohdaten von einer beliebigen Datenbank bzw. Datenquelle akquiriert. Die Daten werden durch den Connector oder einen Datenextractor(Textmining) so transformiert, das sie im Datenmodell gespeichert werden können. Desweiteren bietet jeder Datenextractor die Funktionalität Metriken auf Basis der Rohdaten zu erstellen. Für die

Visualisierung der Daten stellt SONIVIS eine Vielzahl an Optionen bereit. Die Darstellung der Daten wie z.B. Graphen, Tabellen, Statistiken sowie die Interaktionsfläche für den Anwender, dass heißt Buttons, Menüleiste etc. werden nach dem MVC-Pattern gekapselt behandelt, wobei die Controller mit in dieser Subkomponente mit einbebezogen werden, da sie eng mit der grafischen Oberfläche der Applikation verwoben sind. Alle genannten Komponenten werden durch die Erweiterung der jeweiligen abstrakten Klassen im de.sonivis.tool.core- Paket in das Gesamtsytem integriert.



Technologien für die Erweiterbarkeit und Kapselung

Extensions und Extension Points

Da SONIVIS das Eclipse-Framework verwendet ist es möglich eigene Erweiterungen zu schreiben. Diese Erweiterungen werden als Plug-In in die bestehende Applikation mit eingebunden über die Extensions, die Sonivis zur Verfügung stellt, und den Extensions Points. Dieses Konzept ermöglicht eine Modularisierung jedes Softwareproduktes, sodass eine "lose Bindung " unter den einzelnen Systemkomponenten vorliegt, sodass auch bei Veränderungen von Komponenten den unveränderten Teil des Gesamtsystems betrifft. Jede erweiternde Komponente wird in das System mit eingebunden, indem sie an den bestehenden Extension Point ansetzt, der meistens als Proxy-Pattern durch ein Interface realisiert wird, das alle Extensions implementieren sowie dessen Proxys, die den Zugriff von außen auf das Extensionobjekt durch die gemeinsame Schnittstelle gewährleisten. Dieser Mechanismus ist nach dem Prinzip metaphorisch zu verdeutlichen. Die Stehlampe(Extension) muss den passenden Stecker(Extension Point) besitzen um funktionsfähig im System Wohnung eingebettet zu sein.

Datenquellenunabhängigkeit

Durch das verwendete DAO-Pattern ist es möglich verschiedene Datenquellen für die Akquierung der Daten zu verwenden. Bei dem DAO-Pattern wird eine Schnitsttelle implementiert, die Methoden definiert für das Abrufen der Daten. Die Klasse, die die Daten von der Datenbank abruft per SQL-Statements oder die Methoden verwendet zum Auslesen von Dateien, implementiert dieses Interface. Objekte, die Daten nun abrufen wollen, bekommen nun die Daten, die das Interface bereitstellt von dieser Klasse. Dadurch ist immer gewährleistet, dass Geschäftslogik und Datenlogik separiert sind. Bei Änderung der Datenquelle müssen entweder die Methoden der Klasse geändert werden, die die Daten von der Datequelle abruft oder man verwendet eine Factory, die die richtige Klasse, die das DAOInterface implementiert hat, für die passende Datenquelle wählt.

Grundsätliche Struktur – und Entwurfsprinzipien der einzelnen Pakete

SONIVIS Datamodel

Das Datenmodell umfasst ein Set von Objekten, dem InformationSpace, der eindeutig spezifiziert ist,. Dieses Set wird in 2 Teile geteilt und enthält sowohl den "Realteil" und den "Presentationsteil". Beide Teile sind eng miteinander verbunden. Der Realteil umfasst alle Entitäten und Relationen, auf denen das Netzwerk aufbaut. Ein Beispiel für eine Entiät ist ein Actor z.B ein Leser oder ein Content Element z.B eine Seite. Ein Beispiel einer Relation ist die Actor-Content Relation z.B die read Relation, ein Actor hat ein Dokument gelesen. Alle Objekte des Realenteils sind Objekte der generalisierenden Klasse InformationSpaceItem, dessen Objekte eindeutig spezifiziert und einmalig im InformationSpace auftreten.Durch die Abstraktion der realen Objekte als InformationSpaceItem wird die Schnittstelle zum Presentationsteil realisiert. Der Presentationsteil umfasst die abstrakte Darstellung als Graph, der aus Knoten und Kanten besteht. Jeder Graph und jede Graphkomponente(GraphComponent) ist an den InforamtionSpace gebunden als GraphItem und ist eindeutig. Jedem InformationSpaceItem Objekt kann mit einem GraphItem "verknüpft" werden, so dass eine "Brücke" zwischen "Realteil" und "Presentationsteil" entsteht.

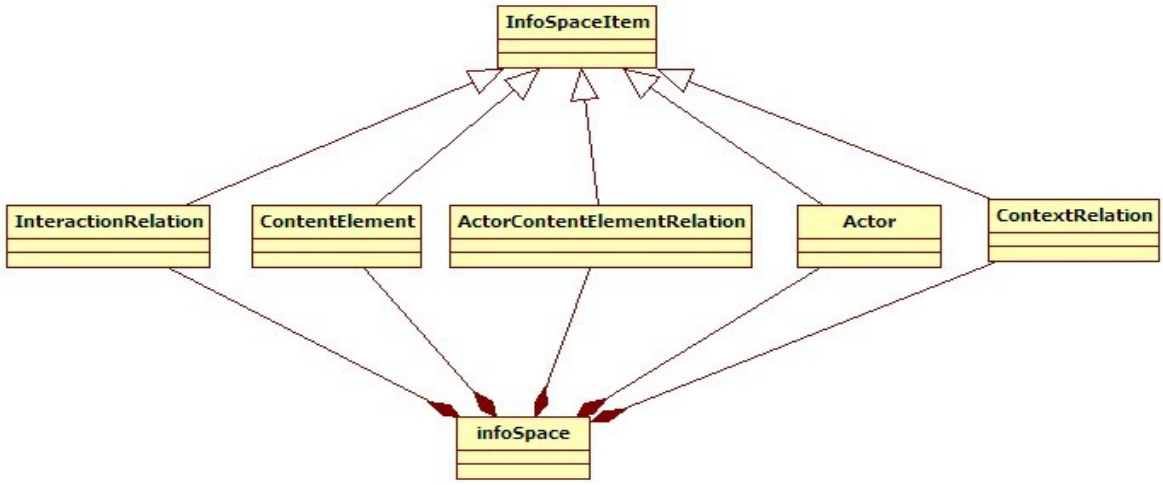


Abbildung 1: Struktur des "Realteils"(aus Übersichtsgründen wurde die Propertyklasse vernachlässigt)

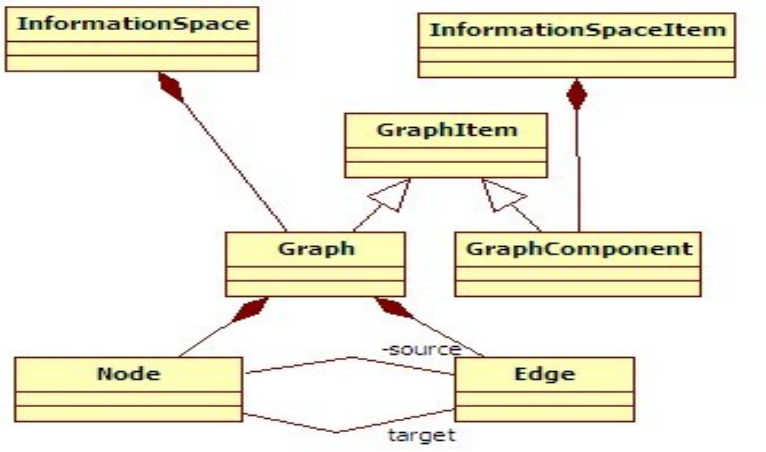


Abbildung 2: Strukur des "Presentationsteil"

Erweiterungsoptionen

SONIVIS bietet die Möglichkeit Erweiterungen zu entwickeln, da alle neu definierten Relationen oder BasicItems die abstrakte InforamtionSpaceItem Klasse nur erweitern müssen oder vorhandene BasicItems bzw. Relations erweitern. So wurde z.B ein Stub gesetzt für Human als Erweiterung des Actor oder eine Erweiterung der Relation ActorContent-Element Relation die Created heißt.

Datenanbindung – Connector

Der Connector ist ein PlugIn für SONIVIS, dessen Aufgabe es ist Daten von einer Datenqulle zu extrahieren und in das SONIVIS-Daenmodell zu speichern. Deweiteren wird jeder spezielle Aspekt der vorhandenen Daten mit berücksüchtigt durch die Berechnung bestimmter Mertriken. Da es sich bei Datenakquierung um ein sehr komplexes Problem handelt, sind 3 Subkomponenten implementiert, die dieses Problem lösen.
Ein Beispiel eines Connectors wird an dem MediaWikiConnector beschrieben.

Extractor

Diese Komponente extrahiert die Daten von einer beliebigen Datenquelle, eine Datei, eine Datenbank etc. und speichert die Daten in einer Datenbank mit der die Applikation arbeiten kann.Diese Extraktion erfolgt beim MediaWikiConnector über die gegebene MediaWikiAPI, die mittels Query ein Ergebnis liefert.

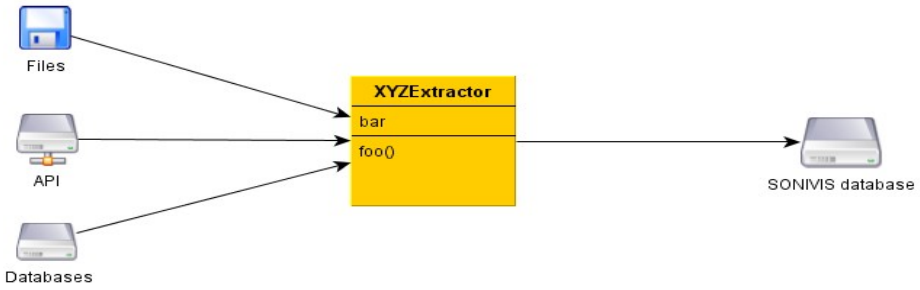


Abbildung 3: Prinzip des Extractor(Quelle:http://sonivis.org/wiki)

Transformer

Der Transformer wandelt die akquierten Daten in Objekte um, die in das Datenmodell passen, dass heißt in InformationSpaceItems bzw. In GraphItems.

DataModel Extensions

Durch die gegebene Verbindung mittels der Extensionpoints werden diese Objekte im Datenmodell als das definierte Objekt, Basicitem, FundementalRelation etc. im InformationSpace gespeichert.

Metriken

Zusätzlich zu der Speicherung der akquierierten Daten im Datenmodell werden Metriken für diese berechnet. So lassen sich die Gesamtanzahl der Relationen oder der BasicItems im Netzwerk ermitteln z.B. die Gesamtanzahl aller Kategorien oder Benutzer.
Die Generierung der Metriken ist im selben Paket, da sie eng mit den Rohadaten zusammenhängen, so lassen sich über einfache Sql-Statements Kardinalitäten von Mengen schnell ermitteln. Des Weiteren sind diese Metriken themenspezifisch, so dass es die optimale Lösung ist, diese bei den spezifischen Connector ermitteln zu lassen.

Implementierung neuer Connectors

Sonivis bietet die Option, neue Connectors zu implementieren, indem jede Komponete mit der dazugehörigen abstrakten Klasse erweitert wird. Durch die DAO Anbindung können die Daten in das Datenmodell gespeichert werden. Desweiteren können neue Metriken definiert werden, indem man die abstrakten Klassen des MetrikSystem im Core-Datenpaket überschreibt.

Grafische Oberfläche des Tool und Visualisierung der Daten

Diese Schicht umfasst sowohl die Implementierung der grafischen Oberfläche mit ihren Menüleisten, sowie die Visualisierung des Graphen mit allen seinen Features wie z.B das Clustering. Des Weiteren werden zum Graphen Zusatzinformationen angezeigt. SONIVIS hat für die strukturelle Einteilung verschiedene Views implementiert, wobei jede View seine eigenen Controller besitzt. Bestehende Views sind:

Characteristic View

Dieses Fenster ist eine Tabelle, in der die spezifischen Metriken zum visuellen Netzwerk presentiert werden.

Statistics View

In diesem Fenster sind Diagramme aufzufinden, die eine Auswertung der Relationen des Netzwerkes darstellen z.B. Eine Auswertung des WikiMedia wäre die Auswertung der Relation zwischen Autor und Artikel, wobei auf der Abzisse die Seiten aufgelistet sind und auf der Ordinate die Anzahl der Autoren pro Seite.

NetworkView

Der Anwender hat hier die Möglichkeit einen Netzwerktyp zu wählen, Collaboration Network, Wiki-Link Network und Dynamic Collaboration Network.

NodeProperty

Es werden die Metriken eines gewählten Knoten im Graphen in einer Tabelle angezeigt.

GraphControlView

Diese Komponente ist das Herzstück der Visualisierung, da hier der Graph auf einem Display dargestellt wird. Desweiteren hat der Anwender die Möglichkeit verschiedene Layouts zu wählen und die Option nach bestimmten Kriterien die Ansicht der Graphen zu reduzieren, z.B. Anzeige aller Knoten mit Grad>5, Label einblenden/ausblenden.

ClusterView

Neben der reinen Visualisierung des Graphen, ist es möglich den Graphen zu Clustern.

Visualisierung des Graphen im Detail

Die allgemeine Erzeugung einer Visualisierung eines Graphen erfolgt in mehrereren Schritten.

1.Schritt:

- Mapping der SONIVIS-Daten und der Prefuse spezifischen Daten Node und Edge

2.Schritt:

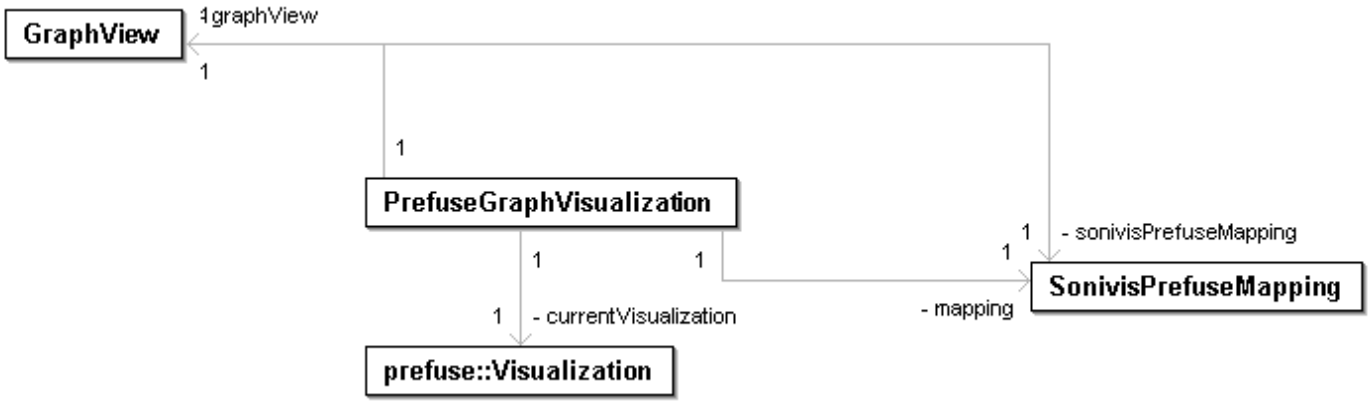
- Erzeugen der PrefuseGraphVisualization, die die Graphinstanz hält mit den gemappten Graphkomponenten, eine Visualizationinstanz sowie eine eingehüllte GraphView-Instanz

3.Schritt:

- Instanzieren des GraphView-Objektes, das ein Panel erweitert und das Display enthält auf dem der Graph dargestellt ist und ein Visualizationobjekt erzeugt mit der Startkonfiguration und somit die Visualization registriert für alle Actions die auf dem Graphen operieren

4.Schritt:

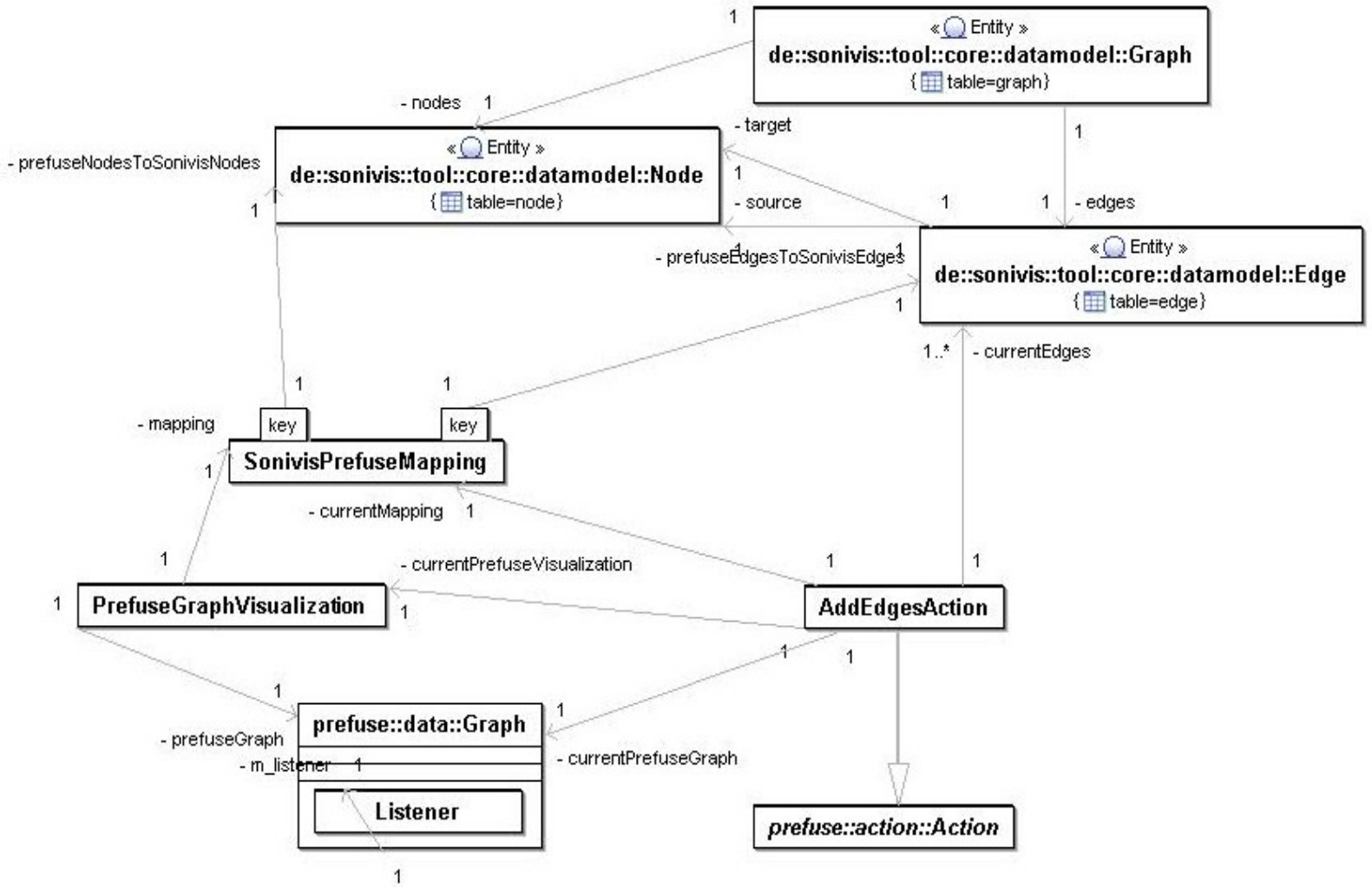
- Visualization-Objekt der PrefuseGraphVisualization auf Visualization der GraphView-Klasse setzen
- Frame erzeugen und GraphView-Objekt, das ein Panel erweitert, auf das Frame setzen



Klasse:de.sonivis.tool.view.prefuse . SonivisPrefuseMapping

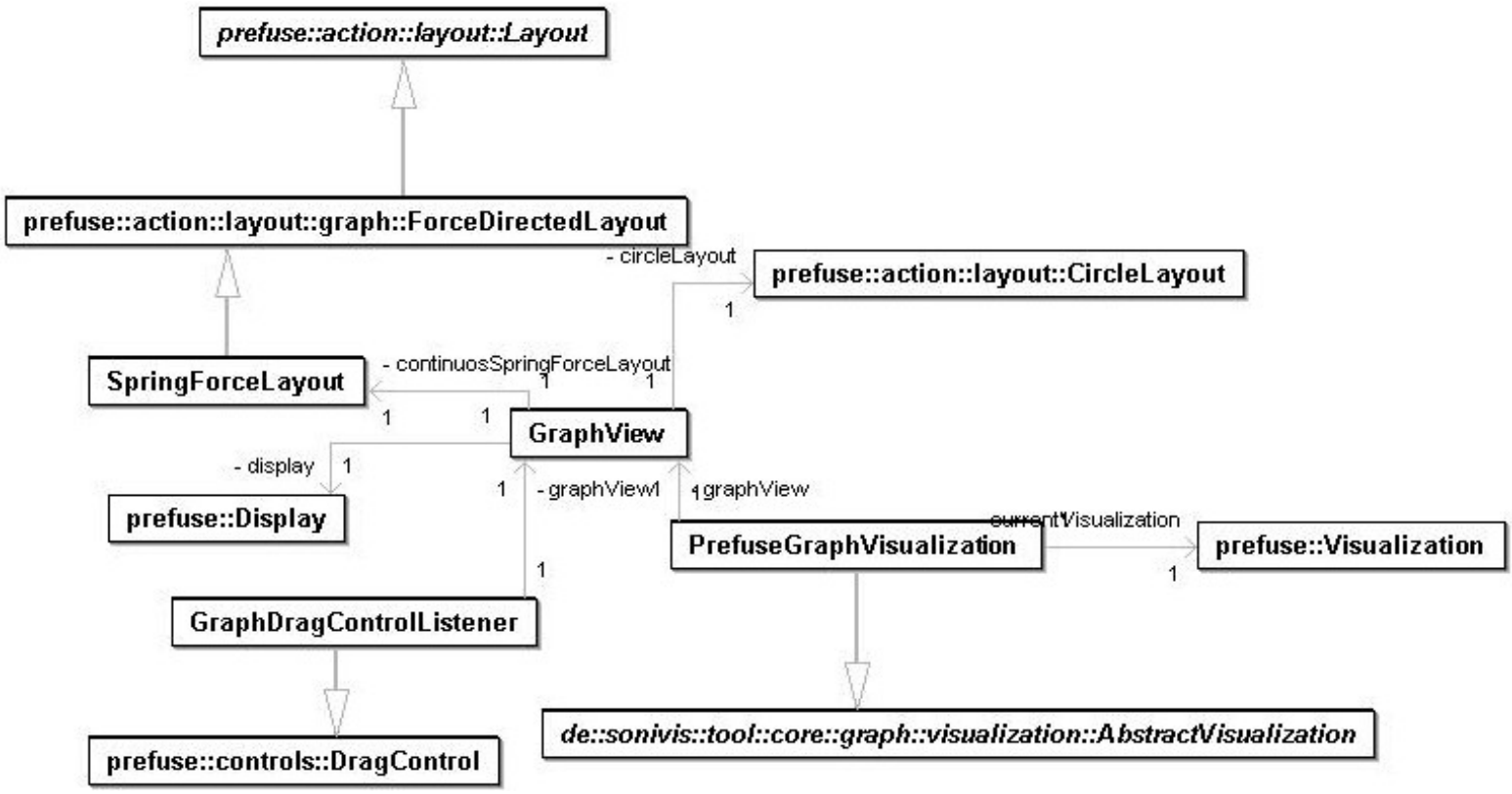
Diese Klasse ist die Schnitstelle zwischen dem SONIVIS- Datenmodell und der Prefuse Speicherstruktur, als Graph. Diese Klasse kann durch ihre Methodenaufrufe für die dazugehörigen Graphkomponenten Kanten und Knoten dem Graphen Hinzufügen bzw. Entfernen, sowohl im Datenmodell als auch im Prefuse-Graphen. Alle Methoden zum Hinzufügen oder Löschen von Graphkomponenten im Datenmodell bzw. im Prefuse-Graphen werden gekapselt durch die jeweilige Klasse realisiert. Alle diese Klassen z.B AddEdgeAction greifen auf die Hashmaps der Klasse SonivisPrefuseMapping zu und aktualisiert daraufhin die Prefuse-Graph-Instanz der PrefuseVisualizationKlasse sowie das Datenmodell. Alle diese Actions überschreiben die abstrakte Action-Klasse von Prefuse, uzm eine Schnittstelle zur Visualisierung zu realisieren. Diese Actions mussten neu implemtiert werden, da Prefuse nicht das Mapping von einem anderen Datenmodell vorsieht, als Graph, Tree oder Table, die prefusespezifische Speicherstrukturen darstellen.

Als exemplarisches Beispiel verdeutlicht das untenstehende Klassendiagramm das Mapping zwischen Datenmodell und Prefuse-Graph, anhand der AddEdgeAction. Methoden und Klassenvariablen wurden aus Interesse an Ihnen, dem Leser, weggelassen.



Klasse:de.sonivis.tool.view.prefuse . SonivisPrefuseVisualization
Diese Klasse ist die Wrapper -Klasse für die Instanz der GraphView-Klasse. Für jede Veränderung des Datenmodells werden durch die Methoden dieser Klasse die jeweilige Action, die in einer eigenen Klasse implementiert ist(z.B. AddEdgeAction), der Visualisierung hinzugefügt und sofort ausgeführt, sodass der visualisierte Graph sich danach ausrichtet.

Klasse: de.sonivis.tool.view.prefuse.GraphView
Es wird eine Instanz einer prefuseVisualization initialisiert, für die die Default-Actions auf dem Graphen hinzugefügt sind, dass heißt Kantenfarbe, Knotenlabels, Knotenfarbe etc.. Des Weiteren werden der Visualisierungsinstanz die Layouts beigefügt, die in separierten Klassen für das jeweilige Layout implementiert sind. Mithilfe der konfigurierten Visualisierung wird das PrefuseDisplay instantiiert, das wiederum dem Panel hinzugefügt, das die Klasse überschreibt. Des Weiteren werden dem Display verschiedene Controller beigefügt, um mit den Graph zu navigieren.
Es werden „layout settings" angelegt. Abhängig von der Änderungen der Einstellungen und Graph-Darstellungsparameters, die der Nutzer des Programms macht, wird die Graph-View im Display aktualisiert. Export von aktuell angezeigten Graphen ist möglich als PNG-Image.



Klasse: de.sonivis.tool.view.graphcontrol.DisplayLayoutView
Diese Klasse dient zur „layout settings"-Darstellung im Graph-Browser. Hier werden Default-Werte für „layout settings" initialisiert und Elemente für Manipulation und Änderung vom Display und von den Einstellungen erstellt.
SONIVIS unterstützt folgende „layout settings":

- Tension
- Min. Länge and Edge Masse-Faktor
- Animation
- NodeLabels-Anzeige
- EdgeLabels-Anzeige
- Clustering usw.

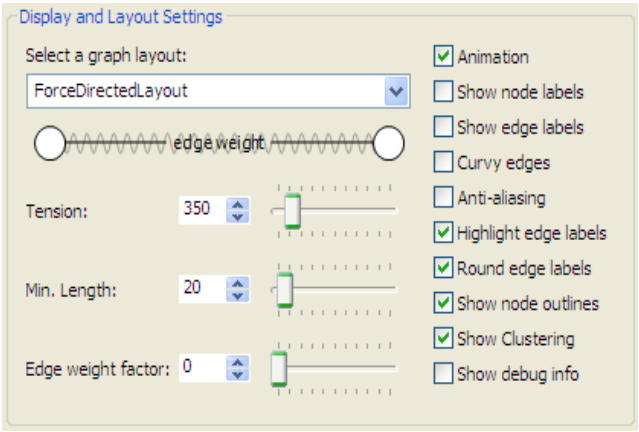


Abbildung 5: Display and Layout Settings (Quelle:http://sonivis.org/wiki)

Clustern des Graphen im Detail

SONIVIS ermöglicht Clustern vom Graphen unter Verwendung von zwei Methoden: partion und hierarchical.
Partition- Methode representiert Dekomposition von einem Netzwerk unter Verwendung von:

- k-means-Methode
- PAM-Methode.

Hierarchical - Methode representiert Dekomposition von einem Netzwerk unter Verwendung von:

- Single linkage algorithm
- Complete linkage algorithm
- Average linkage algorithm
- Ward .

Das Clustern ist im package de.sonivis.tool.view.clustering realisiert. Die allgemeine Erzeugung des Clusters eines Graphen erfolgt in mehrereren Schritten.

1.Schritt:

- Auswahl von den Startpunkten fürs Clustern

2.Schritt:

- Berechnung von Cluster und Einschränkungen dafür

3.Schritt:

- Visualisierung des Clusters und Anzeige von Cluster-Information

Allgemeines zum Einsatz von Prefuse

SONIVIS hat Prefuse genutzt, um die Daten ihres Datenmodells zu visualisieren, dabei nutzen sie die spezifischen Speicherstrukturen der DataTable Schicht um die Daten als Knoten und Kanten in der prefusespezifischen Speicherstruktur Graph zu speichern. Der Graph ist notwendig, da Sonivis die Visualizationklasse von Prefuse nutzt die den Graphen benötigt für die Erstellung der VisualItems, die später auf dem Display als Kanten und Knoten angezeigt werden. SONIVIS nutzt für die Darstellungen des Graphen, die Layouts von Prefuse bzw. erweitert sie. Um die Daten vom Datenmodell zum Graphen zu mappen sind eine Vielzahl von Klassen implementiert, die die Action Klasse von Prefuse überschreiben. Des Weiteren sind Action-Klassen implementiert, die verschiedene Metriken grafisch visualisieren. Für die Interaktion durch den Anwender mit den Graphen werden die Controller von Prefuse verwendet und erweitert.

Umsetzung des IVRM

Da SONIVIS nach eigener Aussage sich am MVC Pattern orientiert, ist eine strikte Einhaltung des IVRM nicht immer gewährleistet, da z.B die Klasse der Visualisierung Daten des Display hält durch ein Objekt der Klasse Graphview(oben beschrieben), so enthält die Klasse der Visualisierung nicht nur Daten für die Darstellung des Graphen, sonder auch das Display selber. Da aber der Umfang der Datengenerierung einen Großteil der Applikation in Anspruch nimmt, ist es angemessen alle Visualisierungen, seien es Statistiken Graphdarstellung mitsamt den Darstellungsdaten, in einem Superpaket view einzuordnen. Diese Einordnung entspricht dem MVC-Pattern.