Entwurfsbeschreibung

1. Allgemeines

Netzwerke gemeinsam auftretender Krankheiten sog. Komorbiditäten können als Graph in einem Human Disease Network dargestellt werden. Das vorliegende Projekt hat zum Ziel eine auf Java basierende Desktop-Applikation zum Verarbeiten der Rohdaten, Darstellung und Exploration solcher Graphen bereitzustellen. Verwendung findet dabei das Toolkit Prefuse.

2. Produktübersicht

Der Comorbidity-Viewer in vorliegender prototypischen Version ermöglicht dem Nutzer eine Datei einzulesen, die über Tabulatoren getrennt in drei Spalten die Rohdaten für das darzustellende HDN enthält. Dabei soll über die ersten beiden Spalten die Knoten identifiziert werden und in der dritten das Kantengewicht vermerkt sein. Dem Nutzer werden alle vorhandenen Knoten in einer Liste präsentiert und der gesamte Graph dargestellt. Durch die Auswahl eines Knotens in der Liste wird dieser im Graphen markiert. Ferner kann der Nutzer Knoten verschieben, zoomen und den Standort des Graphen verschieben.

3. grundsätzliche Struktur- und Entwurfsprinzipien für das Gesamtsystem

Prefuse beruht auf dem „information visualization reference model“ Architekturkonzept. Dieses wurde speziell für Applikationen zur Darstellung komplexer Sachverhalte entwickelt. Prefuse ist dem folgend in 4 große Pakete gegliedert:

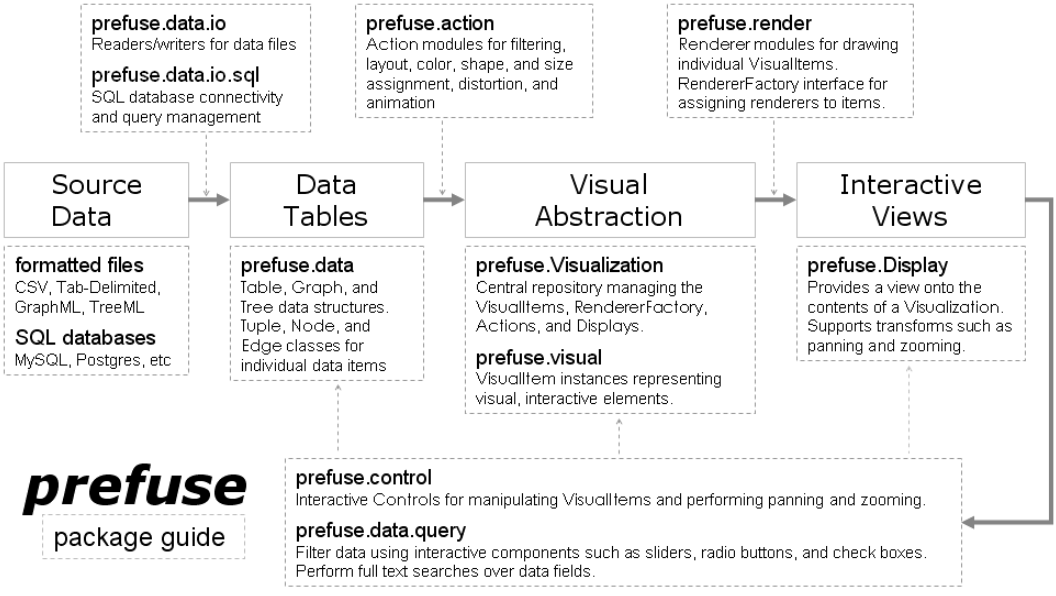

Abbildung 1: Quelle: Prefuse documentation:
http://prefuse.org/doc/manual/introduction/structure/ (Stand 07.05.2010)

Dem Designkonzept von Prefuse folgend, sollte sich jede Applikation, die das Toolkit verwendet an dessen Aufteilung halten.

4. Grundsätzliche Struktur- und Entwurfsprinzipien der einzelnen Pakete

Dem entsprechend umfasst die vorliegende Implementierung eines Comorbidity-Viewer folgende Pakete mit den jeweiligen Verantwortlichkeiten:

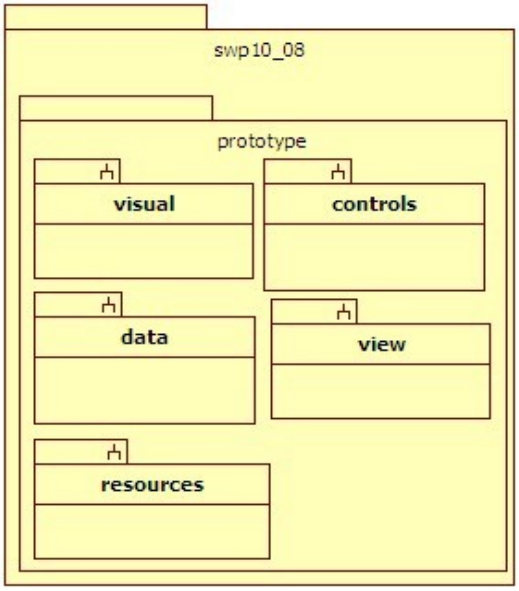

Abbildung 2: Paketdiagramm

- controls *(Steuerung der Anzeige und visual abstraction)*
- view *(Java Swingkomponeneten, Fensterkontrolle und Display für den Graphen)*
- visual *(visuelle Repräsentation der Quelldaten)*
- data *(Einlesen der unterstützten Quellformate in Prefuse Datenformate (tables, graph)* und
- resources *(Verzeichnis mit Icons für die GUI und Quelldateien)*

Wie die Namen der obersten Pakete bereits andeuten, wurde die Funktionalität des vorliegenden Prototypen gemäß dem Prefusearchitekturkonzept aufgeteilt und eigene Implementierungen dem jeweiligen Paket zugeordnet.

Beschreibung von Funktionalität im UML- Klassendiagramm dargestellten Klassen. (s. Abb.3)

- **GraphReader_TST (Data Paket):** Liest einen Graphen aus einer Tab-Separated-Text-Datei (TST) aus. Mittels des DelimitedTextTableReaders von prefuse werden die Daten in einer Tabelle von Prefuse gespeichert. Es werden 3 Spalten ausgelesen, d.h. jede Zeile repräsentiert eine Kante mit Startknoten, Zielknoten und Kantengewicht. Auf Basis der Tabelle werden Kanten- und Knotentabellen erzeugt, die wiederum für die Initialisierung des Graphen(spezielle Speicherstruktur von Prefuse für die Visualisierung als Graph) notwendig sind.
- **SimpleView_Vis (Visual Paket):** Diese Klasse erweitert die Visualization-Klasse von Prefuse. Die implementierte Klasse lässt den Graphen in einem SimpelView-Layout darstellen mit einem Kräftemodell, das Prefuse bereits zur Verfügung stellt durch die Klasse ForceDirectedLayout. Die farbliche Gestaltung der Komponenten des Graphen werden durch die ColorActions realisiert. Um die Komponenten während der Laufzeit bewegen zu können muss der Visualization-Klasse eine Instanz der Klasse RepaintAction hinzugefügt werden. Durch die RendererFactorys von Prefuse werden die Form und die Beschriftung der Knoten realisiert.
- **SimpleView_Controller (Controls Paket):** enthält die konkrete Implementierung der Methoden, die die Controllermethoden von prefuse überschreiben , wie z.B itemClicked.

Desweiteren enthält diese Klasse die Implementierung für alle Interaktionselemente der GUI, die außerhalb des Displays liegen

- **SimpleView_Display (View Paket):** Das Display erweitert die Display-Klasse, auf dem die (abstrakte) Visualisierung des Graphen gezeichnet wird. Die Größe wird festgelegt und einige ControllerListener werden initialisiert, die zum Zoomen und Bewegen des Graphen dienen.

- **SimpleView_GUI (View Paket):** Das ist die GUI der Anwendung. SimpleView_GUI erweitert die Klasse JFrame und beinhaltet einige interaktive Elemente wie Buttons, eine MenuBar, eine ToolBar, eine KnotenListe und das Display, auf dem der Graph gezeichnet wird.

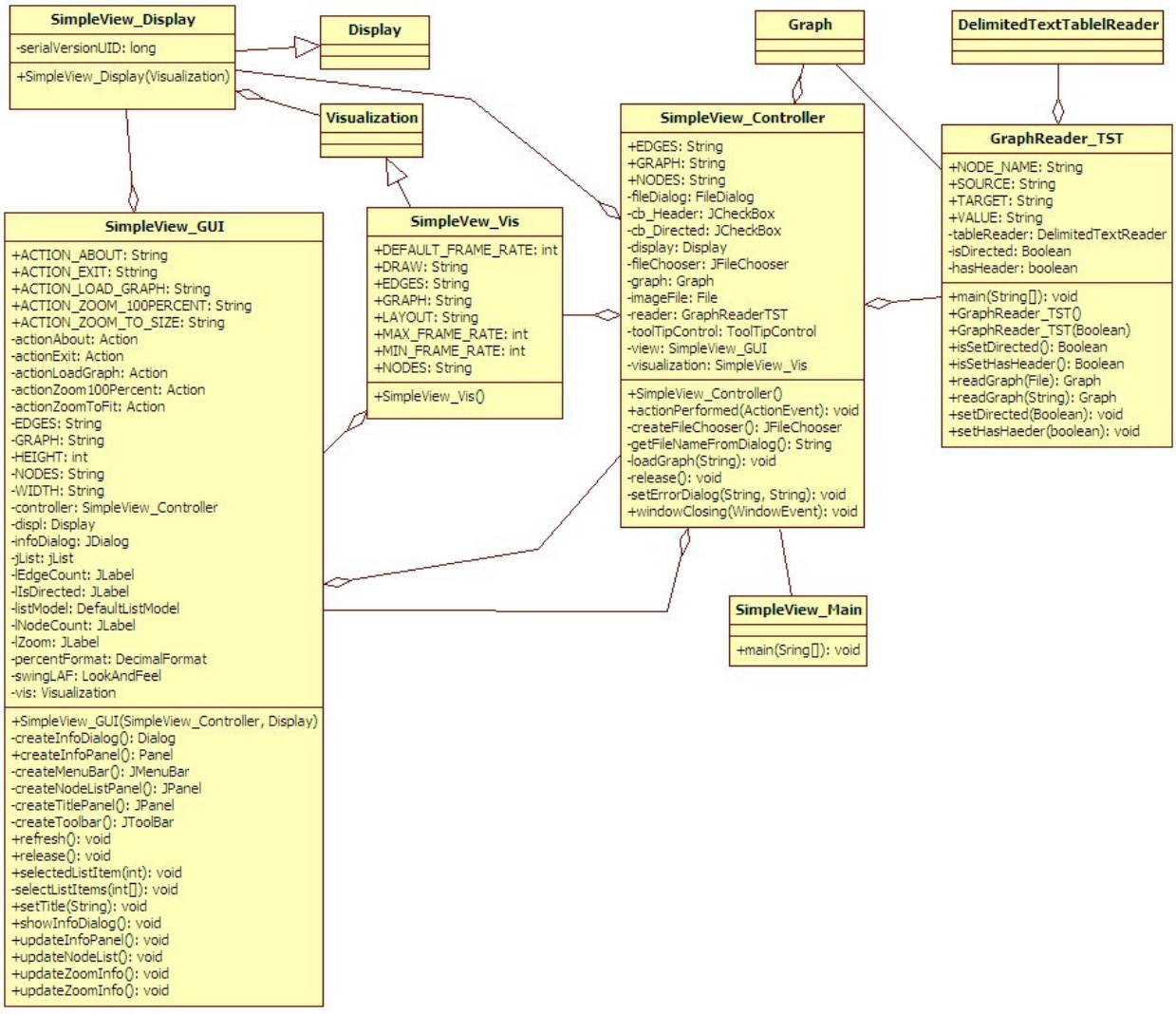

Abbildung 3: Klassendiagramm