Bachelor - Praktikum (Graphenlayout)

Pflichtenheft Rev. 1

Gruppe: $G^{2^{2^2}}$

Stand: 5. Dezember 2006
`http://bp.macrolab.de`

Tutor: Thorsten Volland
Aufraggeber: Michael Eichberg
Gruppenmitglieder:

- Andreas Franke
- Peter Schauss
- Martin Konrad
- Marco Möller

Inhaltsverzeichnis

1. Pflichtenheft

1.1. Einleitung

Dieses Dokument spiegelt unsere Auffassung des Projektes wieder und ist als Angebot an den Auftraggeber zu verstehen. Durch das Gegenlesen können Missverständnisse schon früh im Projekt vermieden werden.

Die Vision enthält einen groben Überblick darüber, wie wir uns das ideale Produckt am Ende der Entwicklung vorstellen.

Ist gibt Aufschluss darüber, was an Software vorhanden ist. Zudem werden deren Mängel und somit der Grund für diese Entwicklung erläutert.

Soll entspricht der Vision im Detail. Hier sind alle Features einzeln aufgeführt und erläutert.

1.2. Vision

Es soll ein Eclipse-Plugin zur Visualisierung von kleinen bis mittleren Graphen entstehen. Die Größe der Graphen sollte dabei etwa 1-50 Knoten umfassen. Der Benutzer soll das Plug-In sowie alle abängige Plug-Ins einfach und schnell von Update-Seiten herunterladen können. Dabei sollten keine Abhängigkeiten zu Plug-Ins bestehen, die nicht frei verfügbar sind.

Über zwei Schnittstellen sollen Graphen definiert werden können. Zum einen über eine Java-API, so dass einfach und schnell aus anderen Java-Applikationen Graphen erstellt werden können, zum anderen über XML-Dateien. Hierbei ist eine vollständige Interaktion zwischen beiden Erzeugungsmethoden möglich, d.h. über die API erzeugte Graphen können im XML-Format gespeichert werden und aus dem XML-Format erzeugte Graphen können über die API manipuliert werden. Außerdem kann der angezeigte Graph als Bilddatei exportiert werden.

Hauptanwendung ist letztendlich die Visualisierung der Graphen über eine Eclipse-View. Die Berechnung des Graphenlayouts kann jederzeit vom Benutzer abgebrochen werden. Bei der Betrachtung von mehreren Graphen auf einmal werden diese überschaubar, übersichtlich angzeigt. Den Knoten können Texte beliebiger Länge zugeordnet werden, die sich auf- und zuklappen lassen. Die Möglichkeit Subgraphen und Mehrfachkanten zu verwenden ist auch vorgesehen. Die, bei Bedarf gerichteten, Kanten sind ebenfalls beschriftbar und zur besseren Unterscheidung farblich markierbar. Ebenso gibt es mehrere Knotentypen, die man verwenden kann. Die Knoten- und Kantentexte lassen sich mit Hilfe von HTML Tags formatieren.

Zusätzlich kann man Skripte an die Knoten und Kanten hängen, die über einen Rechtsklick in den Graphen ausgeführt werden können. Grundbefehle, auf die die Skripte zugreifen können, sind im Plug-In implementiert. Es lassen sich mehrere verschiedene Skriptsprachen verwenden. Große Graphen werden zusätzlich in einem Overview Fenster angezeigt und unterstützen Pan & Zoom Funktionen.

1.3. Ist

Bislang existiert in der Arbeitsgruppe nur ein Plug-In zur Visualisierung von Graphen, das einen Export als Grafik erlaubt. Allerdings stürzt es ab und zu ab. Zudem gibt es manchmal keine Anzeige der Graphen, was sich nur durch einen Eclipse Neustart beheben läßt. Außerdem ist kein XML Dateiformat spezifiziert und es ist nicht in der Lage Funktionen wie Zoom in der GUI auszuführen. Auch fehlt eine Möglichkeit Skripte einzubinden.

1.4. Soll

Das Soll unserers Projektes kann aus der nachfolgenden Tabelle entnommen werden. Hier sind alle gewünschten Features im Detail beschrieben. Die Prioritäten sind als Schulnoten für das Endergebnis zu verstehen:

5 & 4 sind verpflichtende Features

3 optionale Features, die nach Möglichkeit implementiert werden

2 wünschenswerte Features, deren Implementation im Konzept vorgesehen wird

1 gehört zur ultimativen Ausbaustufe und eigentlich nicht erforderlich

Priorität	Beschreibung
	Eclipse Plug-in
5	Das Eclipse 3.2 Plug-In muss unter Windows und Linux mit denselben Funktionalitäten lauffähig sein. Es dürfen keine Abhängigkeiten zu Tools und Bibliotheken existieren, die nicht nicht unter MacOS X ausführbar sind. Das Plug-In muss unter Java 5 kompilierbar sein.
5	Die Installation erfolgt über eine Eclipse Update-Site, dabei dürfen keine Abhängigkeiten zu Tools existieren, die sich nicht mit Hilfe dieser installieren lassen.
5	Die simultane Anzeige mehrerer Graphen, z.B. nebeneinander, sollte möglich sein.
4	Berechnungen sollen abbrechbar sein, falls sie zu lange dauern
3	Die Anzeige der Graphen sollte als Eclipse View erfolgen.
	Visualisierung eines Graphen
5	Knoten können mit einem mehrzeiligen Text beschriftet werden.
5	Als Zeichensatz für den Text eines Knotens sollte UTF-8 (oder UTF-16) unterstützt werden - die Zeichen, die außerhalb des ASCII Zeichensatzes definiert sind, müssen auch unterstützt werden.
5	Die Form und das Layout der Knoten ist variabel. Es lässen sich Farben und die Dicke des Rahmens variieren.
5	Mehrfachverbindungen zwischen zwei Knoten müssen unterstützt werden und auch visuell erkennbar sein
5	Mehrfachverbindungen eines Knotens mit sich selbst müssen erkennbar sein.
5	Gerichtete und ungerichtete Verbindungen müssen unterstützt werden.
4	Es sollten mindestens fünf verschiedene Typen von Verbindungen unterstützt werden, erkennbar durch unterschiedliche Farbe und Dicke der Linien
4	Es sollte möglich sein, Kanten zu beschriften.
4	Es sollten mindestens drei verschiedene Start- und Endknotentypen verfügbar sein.
3	Unterstützung von Subgraphen.
3	Als Knotentext kann HTML formatierter Text verwendet werden.
3	Zusammengesetzte Knoten, d.h. es sollte möglich sein Text nebeneinander zu setzen.
2	Knoten sollten faltbar sein, d.h. der Benutzer sollte in der Lage sein, Knoten mit viel Text zusammenzufalten und dann nur einen Kurzbezeichner zu sehen.
	Interaktion mit dem Graphen
5	Unterstützung für ein Skript, z.B. in JavaScript, welches an einen Knoten angehängt wird
3	Unterstützung für mehr als ein Skript.
3	Filterung von verschiedenen Verbindungen sollte möglich sein, so dass man Verbindungstypen ausschalten kann
3	Es sollte möglich sein, Basisfunktionalitäten vorzudefinieren, auf die dann in den Skripten zugegriffen werden kann. Zum Beispiel eine Funktion die bei gegebenen Methodennamen und Namen der deklarierenden Klasse einen Editor öffnet und zur entsprechenden Methode navigiert.
3	Es sollte möglich sein angezeigte Graphen als Grafiken exportieren zu können.
2	Ein Overview Fenster das bei größeren Graphen anzeigt wo man sich befindet.
1	Unterstützung für mehr als eine Skriptsprache.
1	Es sollte möglich sein an Verbindungen zwischen zwei Knoten Skripte zu hängen.
	Sonstiges Anforderungen
5	Es dürfen keine offensichtlichen lizenzrechtlichen Beschränkungen existieren, die die freie zur Verfügungstellung des Plug-ins behindern.
5	Eine Webseite inkl. Eclipse Update-Site, welche das Plug-in vorstellt und die Verwendung erklärt.

1.5. Use-Cases

Use-Cases sind abstrakte Modelle von Funktionalitäten oder Diensten, welche das System den Benutzern (auch als Akteure bezeichnet) anbietet. Jedes Use-Case beschreibt dabei ein Szenario, wie das System dabei mit dem Akteur interagieren kann um ein gewisses Ziel zu erreichen. Use-Cases betrachten das System als 'black box' und haben daher nur Einsicht auf Dinge, die vom System nach außen dringen.
Hier als Beispiel einen leeren Use-Case, gefolgt von der Definition der einzelnen Felder:

0

Name		
ID		
Status		
Priorität		
Akteur		
Kurzbeschreibung		
Vorbedingung		
Nachbedingung		
	Aktion	Reaktion
Normaler Ablauf		
Alternativer Ablauf		

Name: Der eindeutige Name des Use-Cases, der einen Hinweis auf die Funktion des Use-Cases gibt.

ID: Eine einzigartige ID, die einen Hinweis darauf liefert in welchem Zusammenhang das Use-Case mit den anderen Use-Cases steht, und in welchem Fall es eingesetzt wird (zB API).
Format: UC-[INS|API|GUI]-*

Status: Der Fortschritt der Umsetzung des Use-Cases (Arten: In Planung, In Arbeit, Abgeschloßen)

Priorität: Wie wichtig die Umsetzung des Use-Cases ist (Arten: Hoch, Mittel, Niedrig)

Akteur: Akteur, der dieses Use-Case ausführt. Dies können bei uns folgende sein:

Benutzer Ein Mensch, der mit Hilfe von Maus und Tastatur in Eclipse das Use-Case ausführt.

Programm über API Ein Java Programm, das über die von uns angebotene API Schnitstelle das Use-Case ausführt.

Kurzbeschreibung: Eine kurze Beschreibung, was der eigentliche Sinn des Use-Cases ist. Dient dazu, dass der Leser nicht das komplette Use-Case lesen muss um zu verstehen worum es geht.

Vorbedingung: Die Vorbedingungen geben an, welche Bedingungen erfüllt sein müssen, damit der Benutzer das Use-Cases initiieren kann. Sind diese Bedingungen nicht erfüllt, so funktioniert das Use-Case nicht.

Nachbedingung: Die Nachbedingungen geben an, wie das System nach der Ausführung des Use-Cases aussieht.

Normaler Ablauf: Der detaillierte Ablauf des Use-Cases. In Listenform.

Alternativer Ablauf: Alternative Abläufe des Use-Cases. Auch hier wieder als Liste, wobei auf Punkte aus dem normalen Ablauf verwiesen werden kann.

Aktion: Aktive Einflüße die das System beim Durchlaufen des Use-Cases betreffen.

Reaktion: Das direkte Ergebnis (Nachbedingung) der unter Aktion ausgeführten Schritte.

1.5.1. Installations-Use-Cases

Die Installation unseres Plugins gehört explizit zu unserem Aufgabenbereich. Somit sind auch hier Use-Cases im Folgenden vorhanden.

1

Installation des Plugins		
ID	UC-INS-Install	
Status	In Planung	
Priorität	Hoch	
Akteur	Benutzer	
Kurzbeschreibung	Das Plugin ist über eine Update-Site installierbar.	
Vorbedingung	Die Eclipse-Platform ist in Version 3.2 installiert.	
Nachbedingung	Das Plugin und alle benötigten vorausgesetzten Plugins wurden installiert.	
	Aktion	Reaktion
Normaler Ablauf	1 Der Benutzer verwendet die Update-Routine des Eclipse-Platform, um das Plugin zu installieren.	2 Das Plugin wird mit den auf der Update-Seite angegebenen Parametern installiert.
Alternativer Ablauf		

2

Deinstallation des Plugins		
ID	UC-INS-DeInstall	
Status	In Planung	
Priorität	Hoch	
Akteur	Benutzer	
Kurzbeschreibung	Das Plugin lässt sich mit Eclipsemitteln deinstallieren.	
Vorbedingung	Die Eclipse-Platform ist in Version 3.2 installiert. Das Plugin ist installiert.	
Nachbedingung	Das Plugin und alle dazugehörigen Dateien sind gelöscht.	
	Aktion	Reaktion
Normaler Ablauf	1 Der Benutzer verwendet die Update-Routine des Eclipse-Platform um das Plugin zu deinstallieren.	2 Das Plugin und alle dazugehörigen Dateien werden gelöscht.
Alternativer Ablauf		

1.5.2. API-Use-Cases

Ein Teil der Aufgabenstellung fordert, dass unser Programm eine API bereitstellen soll. Die folgenden Use-Cases beschreiben, wie die Interaktion dabei geschehen soll. Vorerst ist kein Löschen von Objekten wie z.B. Knoten und Kanten vorgesehen. Da die Graphen sowieso von anderen Programmen über die API erstellt werden, ist damit auch kein Zusatzaufwand für den Benutzer verbunden.

3

Erzeugen des Graphen		
ID	UC-API-CreateGraph	
Status	In Planung	
Priorität	Hoch	
Akteur	Programm über API	
Kurzbeschreibung	Erzeugen eines Graph-Objekts	
Vorbedingung	Installiertes Plugin.	
Nachbedingung	Ein Graph-Objekt wurde erstellt.	
	Aktion	Reaktion
Normaler Ablauf	1 Der Graph wird über den Konstruktor erzeugt.	2 Graph-Objekt wird erzeugt und zurückgegeben.
Alternativer Ablauf		

4

Erzeugen eines Knotens		
ID	UC-API-CreateNode	
Status	In Planung	
Priorität	Hoch	
Akteur	Programm über API	
Kurzbeschreibung	Erzeugen eines Knoten-Objekts	
Vorbedingung	Installiertes Plugin.	
Nachbedingung	Ein Knoten-Objekt wurde erstellt.	
	Aktion	Reaktion
Normaler Ablauf	1 Der Knoten wird über den Konstruktor erzeugt. Parameter dabei sind: Beschriftung, Farbe, Typ, Inhalt. Inhalt ist entweder ein weiterer Graph (Subgraph) oder ein ausführlicher Text (zum Ausklappen).	2 Knoten-Objekt wird erzeugt und zurückgegeben.
Alternativer Ablauf		

5

Erzeugen einer Kante		
ID	UC-API-CreateEdge	
Status	In Planung	
Priorität	Hoch	
Akteur	Programm über API	
Kurzbeschreibung	Erzeugen eines Kanten-Objekts	
Vorbedingung	Installiertes Plugin. Es wurden Knotenobjekte erzeugt.	
Nachbedingung	Ein Kanten-Objekt wurde erstellt.	
	Aktion	Reaktion
Normaler Ablauf	1 Die Kante wird über den Konstruktor erzeugt. Parameter dabei sind: Beschriftung, Farbe, Typ, Gerichtet.	2 Kanten-Objekt wird erzeugt und zurückgegeben.
Alternativer Ablauf		

6

Erzeugen einer Skriptumgebung		
ID	UC-API-CreateScriptEnv	
Status	In Planung	
Priorität	Hoch	
Akteur	Programm über API	
Kurzbeschreibung	Erzeugen einer Umgebung fuer ein Skript	
Vorbedingung	Installiertes Plugin.	
Nachbedingung	Es wurde eine Skriptumgebung erstellt.	
	Aktion	Reaktion
Normaler Ablauf	1 Die Umgebung wird über den Konstruktor erzeugt. Parameter dabei sind: Skritsprache, Pfad, Code.	2 Umgebungs-Objekt wird erzeugt und zurückgegeben.
Alternativer Ablauf		

7

Erzeugen eines Skriptes		
ID	UC-API-CreateScript	
Status	In Planung	
Priorität	Hoch	
Akteur	Programm über API	
Kurzbeschreibung	Anlegen eines Skripts	
Vorbedingung	Installiertes Plugin und es wurde eine Umgebung erstellt.	
Nachbedingung	Es wurde ein Skript angelegt.	
	Aktion	Reaktion
Normaler Ablauf	1 Das Skript wird über den Konstruktor erzeugt. Parameter dabei sind: Beschriftung, Umgebung, Code.	2 Skript-Objekt wird erzeugt und zurückgegeben.
Alternativer Ablauf		

8

Hinzufügen eines Knotens		
ID	UC-API-AddNode	
Status	In Planung	
Priorität	Hoch	
Akteur	Programm über API	
Kurzbeschreibung	Knoten in Graph einfügen	
Vorbedingung	Installiertes Plugin. Es muss ein Graph und ein Knoten-Objekt erzeugt worden sein	
Nachbedingung	Ein Knoten wurde hinzugefügt.	
	Aktion	Reaktion
Normaler Ablauf	1 Die 'addNode' Funktion des Graph-Objekts wird aufgerufen.	2 Graph-Objekt hat einen Knoten mehr.
Alternativer Ablauf		

9

Hinzufügen einer Kante		
ID	UC-API-AddEdge	
Status	In Planung	
Priorität	Hoch	
Akteur	Programm über API	
Kurzbeschreibung	Kante in Graph einfügen	
Vorbedingung	Installiertes Plugin. Es müssen zwei Koten und ein Graph-Objekte erzeugt worden sein.	
Nachbedingung	Der Graph enthält zusätzlich die eingefügte Kante	
	Aktion	Reaktion
Normaler Ablauf	1 Über die Java-API wird die Funktion 'addEdge' aufgerufen mit Ursprungs- und Zielknoten als Parameter.	2 Die Existenz der beiden Knoten wird geprüft. 3 Die Kante wird im Graph-Objekt eingefügt.
Alternativer Ablauf		

10

Binden eines Skriptes an Knoten oder Kante		
ID	UC-API-AddSkript	
Status	In Planung	
Priorität	Hoch	
Akteur	Programm über API	
Kurzbeschreibung	Knoten oder Kante mit Skript hinzufügen	
Vorbedingung	Installiertes Plugin. Es muss ein Skript und ein Knoten- bzw. Kanten-Objekt erzeugt worden sein.	
Nachbedingung	Das Skipt wurde mit dem Objekt verbunden.	
	Aktion	Reaktion
Normaler Ablauf	1 Die 'addSkript' Funktion des Knoten- oder Kanten-Objekts wird aufgerufen.	2 Knoten- oder Kanten-Objekt hat eine Skript mehr in seinem Kontextmenü.
Alternativer Ablauf		

11

Setzen eines Filters		
ID	UC-API-SetFilter	
Status	In Planung	
Priorität	Mittel	
Akteur	Programm über API	
Kurzbeschreibung	Ein Filter-Objekt wird erzeugt und abgelegt.	
Vorbedingung	Installiertes Plugin.	
Nachbedingung	Das Filter-Objekt wurde erzeugt und gespeichert.	
	Aktion	Reaktion
Normaler Ablauf	1 Das Filter-Objekt wurde über seinen Konstruktor mit entsprechenden Parametern erzeugt.	2 Das Filter-Objekt wird erzeugt und gespeichert.
Alternativer Ablauf		

12

Rücksetzen eines Filters		
ID	UC-API-ClearFilter	
Status	In Planung	
Priorität	Mittel	
Akteur	Programm über API	
Kurzbeschreibung	Der Filter wird zurückgesetzt.	
Vorbedingung	Installiertes Plugin.	
Nachbedingung	Es ist kein Filter gesetzt.	
	Aktion	Reaktion
Normaler Ablauf	1 Die Funktion clearFilter() wird aufgerufen.	2 Der Filter wird zurückgesetzt.
Alternativer Ablauf		

13

Berechnung des Graphenlayouts		
ID	UC-API-Layout	
Status	In Planung	
Priorität	Hoch	
Akteur	Programm über API	
Kurzbeschreibung	Das Layout des Graphen wird basierend auf der vorliegenden Definition berechnet	
Vorbedingung	Installiertes Plugin. Ein Graph ist vollständig definiert	
Nachbedingung	Das Layout des Graphen ist bekannt	
	Aktion	Reaktion
Normaler Ablauf	1 Ein Algorithmus für das Layout wird in einem neuen Thread gestartet.	2 Das Layout wird unter berücksichtigung von Filtern und ein- bzw. ausgeklappten Knoten berechnet. 3 Das fertige Layout wird intern gespeichert.
Alternativer Ablauf		3b Bei fehlerhaft definiertem Graphen wird eine Fehlermeldung ausgegeben.

14

Abbruch der Berechnung des Graphenlayouts		
ID	UC-API-LayoutAbort	
Status	In Planung	
Priorität	Hoch	
Akteur	Programm über API	
Kurzbeschreibung	Die Berechung des Graphenlayouts wird abgebrochen.	
Vorbedingung	Installiertes Plugin. Es findet im Moment eine Berechnung eines Graphenlayouts statt.	
Nachbedingung	Es findet keine Berechnung mehr statt.	
	Aktion	Reaktion
Normaler Ablauf	1 Abbruchfunktion wird aufgerufen.	2 Der Berechnungsthread wird beendet und alle Layoutdaten werden verworfen.
Alternativer Ablauf		

15

Zeichnen von Graphen in GUI		
ID	UC-API-Draw	
Status	In Planung	
Priorität	Hoch	
Akteur	Programm über API	
Kurzbeschreibung	Das berechnete Graphenlayout wird angezeigt.	
Vorbedingung	Installiertes Plugin. Es ist ein Graph definiert und es hat eine dazugehörige Layoutberechnung erfolgreich stattgefunden.	
Nachbedingung	Ein Fenster mit den angezeigtem Graphen ist geöffnet.	
	Aktion	Reaktion
Normaler Ablauf	1 Zeichenfunktion wird aufgerufen.	2 Der Graph wird gezeichnet.
Alternativer Ablauf		

16

Exportieren von Graphen nach SVG		
ID	UC-API-Export	
Status	In Planung	
Priorität	Hoch	
Akteur	Programm über API	
Kurzbeschreibung	Das berechnete Graphenlayout wird in SVG Datei gespeichert.	
Vorbedingung	Installiertes Plugin. Es ist ein Graph definiert und es hat eine dazugehörige Layoutberechnung erfolgreich stattgefunden.	
Nachbedingung	Es existiert eine SVG Datei die dem Graphenlayout entspricht.	
	Aktion	Reaktion
Normaler Ablauf	1 Exportfunktion wird unter Angabe des Dateinamens aufgerufen.	2 Der Graph wird in die entsprechnede Datei exportiert.
Alternativer Ablauf		

17

Ausführen eines Skripts		
ID	UC-API-RunScript	
Status	In Planung	
Priorität	Hoch	
Akteur	Programm über API	
Kurzbeschreibung	Ein Skript-Objekt wird in seiner Umgebung ausgeführt.	
Vorbedingung	Installiertes Plugin. Es ist ein Skript-Objekt definiert.	
Nachbedingung	Das Skript-Objekt wurde erfolgreich ausgeführt.	
	Aktion	Reaktion
Normaler Ablauf	1 Die 'run()' Funktion des Skriptobjekts wird aufgerufen.	2 Ein Interpreter für die entsprechende Sprache führt das Skript in seiner Umgebung aus.
Alternativer Ablauf		

18

Speichern eines Graphen		
ID	UC-API-Save	
Status	In Planung	
Priorität	Hoch	
Akteur	Programm über API	
Kurzbeschreibung	Der Graph inklusive der Skripte, allerdings ohne Layout, wird gespeichert.	
Vorbedingung	Installiertes Plugin. Es ist ein Graph definiert.	
Nachbedingung	Der Graph inklusive der Skripte, allerdings ohne Layout, ist in die Datei geschrieben worden.	
	Aktion	Reaktion
Normaler Ablauf	1 Die Speichern-Funktion wird unter Angabe des Dateinamens gestartet.	2 Der Graph inklusive der Skripte, allerdings ohne Layout, wird in die Datei geschrieben.
Alternativer Ablauf		

19

Laden eines Graphen		
ID	UC-API-Load	
Status	In Planung	
Priorität	Hoch	
Akteur	Programm über API	
Kurzbeschreibung	Basierend auf einer gespeicherten Graphendefinition werden Objekte erzeugt, die diesen repräsentieren	
Vorbedingung	Installiertes Plugin. Eine Datei im passenden Format zum Laden ist vorhanden.	
Nachbedingung	Der Graph wurde intern erzeugt.	
	Aktion	Reaktion
Normaler Ablauf	1 Die Lade-Funktion wird unter Angabe des Dateinamens gestartet.	2 Die entsprechenden Objekte werden aus der Datei geladen und angelegt.
Alternativer Ablauf		

1.5.3. GUI-Use-Cases

Alle Übrigen Use-Cases beziehen sich auf die Interaktion mittels der graphischen Oberfläche mit unserem Plugin.

20

Zoomen der Ansicht		
ID	UC-GUI-Zoom	
Status	In Planung	
Priorität	Mittel	
Akteur	Benutzer	
Kurzbeschreibung	Die Graphenansicht kann vergrößert und verkleinert werden.	
Vorbedingung	Im Anzeigefenster angezeigter Graph.	
Nachbedingung	vergrößerter bzw. verkleinerter Graph im Anzeigefenster.	
	Aktion	Reaktion
Normaler Ablauf	1 Der Nutzer klickt auf eine leere Position im Graphen und zieht bei gedrückter rechter Maustaste nach oben bzw. unten.	2 Die Größe des Graphens im Fenster ändert sich.
Alternativer Ablauf		

21

Verschieben der Ansicht (‚Pan‘)		
ID	UC-GUI-Pan	
Status	In Planung	
Priorität	Mittel	
Akteur	Benutzer	
Kurzbeschreibung	Die Graphenansicht kann verschoben werden.	
Vorbedingung	Im Anzeigefenster angezeigter Graph.	
Nachbedingung	Graph ist im Anzeigefenster verschoben	
	Aktion	Reaktion
Normaler Ablauf	1 Der Nutzer klickt auf eine leere Position im Graphen und bewegt den Mauszeiger bei gedrückter linker Maustaste.	2 Die Position des Graphens im Fenster ändert sich.
Alternativer Ablauf		

22

Reset Zoom + Pan		
ID	UC-GUI-Reset	
Status	In Planung	
Priorität	Mittel	
Akteur	Benutzer	
Kurzbeschreibung	Die Zoom und Pan Einstellungen werden zurückgesetzt.	
Vorbedingung	Im Anzeigefenster angezeigter Graph.	
Nachbedingung	Der Graph ist vollständig im Anzeigefenster sichtbar.	
	Aktion	Reaktion
Normaler Ablauf	1 Der Nutzer klickt auf den Reset Button.	2 Die original Werte für Zoom und Pan werden geladen und das Fenster entsprechend angepasst.
Alternativer Ablauf		

23

Anzeigen des Graphen-Overviews		
ID	US-GUI-OverviewShow	
Status	In Planung	
Priorität	Niedrig	
Akteur	Benutzer	
Kurzbeschreibung	Der Benutzer kann sich einen Overview des Graphen anzeigen lassen.	
Vorbedingung	Das Graphenlayout wurde berechnet und es wird in der View angezeigt.	
Nachbedingung	Man sieht zusätzlich ein kleines Overview-Fenster mit dem Graph.	
	Aktion	Reaktion
Normaler Ablauf	1 Der Benutzer fordert über die Eclipse-Plattform ein Overview-Fenster als View an.	2 Ein Overview-Fenster wird geöffnet. 3 Der Graph wird im Overview-Fenster so skaliert, dass er komplett sichtbar ist.
Alternativer Ablauf		

24

Ausfalten eines Knoten		
ID	UC-GUI-Unfold	
Status	In Planung	
Priorität	Niedrig	
Akteur	Benutzer	
Kurzbeschreibung	Knoten wird ausgefaltet.	
Vorbedingung	Es wird ein Graph angezeigt mit zusammengefaltetem Knoten.	
Nachbedingung	Der Knoten ist ausgefaltet.	
	Aktion	Reaktion
Normaler Ablauf	1 Der Benutzer klickt das Ausfalten-Symbol auf dem Knoten.	2 Der Knoten wird ausgefaltet. 3 Die Anzeige wird aktualisiert mit UC-API-Draw.
Alternativer Ablauf		

25

Zusammenfalten eines Knoten		
ID	UC-GUI-Fold	
Status	In Planung	
Priorität	niedrig	
Akteur	Benutzer	
Kurzbeschreibung	Knoten wird zusammengefaltet.	
Vorbedingung	Es wird ein Graph angezeigt mit einem ausgefalteten Knoten.	
Nachbedingung	Der Knoten ist zusammmengefaltet.	
	Aktion	Reaktion
Normaler Ablauf	1 Der Benutzer klickt auf das Zusammenfalten-Symbol auf dem Knoten.	2 Der Knoten wird gefaltet. 3 Die Anzeige wird aktualisiert mit UC-API-Draw.
Alternativer Ablauf		

26

Anzeige zweier Graphen nebeneinander		
ID	UC-GUI-Dual	
Status	In Planung	
Priorität	Hoch	
Akteur	Benutzer	
Kurzbeschreibung	Es können zwei Graphen nebeneinander angezeigt werden	
Vorbedingung	Es ist mindestens ein Graph definiert worden.	
Nachbedingung	Zwei Fenster nebeneinander, in denen jeweils ein Graph zu sehen ist.	
	Aktion	Reaktion
Normaler Ablauf	1 Öffnen eines zweiten Anzeigefensters 3 Laden eines Graphen	2 Zweites Fenster wird geöffnet. 4 Graph wird geladen.
Alternativer Ablauf	1b Im geöffneten Graph-Fenster auf den Duplizieren-Button klicken.	

27

Setzen eines Filters		
ID	UC-GUI-SetFilter	
Status	In Planung	
Priorität	Mittel	
Akteur	Benutzer	
Kurzbeschreibung	Nur Kanten eines gewissen Typus werden eingeblendet.	
Vorbedingung	Ein Graph wird angezeigt.	
Nachbedingung	Der Graph wird mit den definierten Kanten-Typen wird angezeigt.	
	Aktion	Reaktion
Normaler Ablauf	1 Im Filtermenü Kanten-Filter setzen.	2 Es wird UC-API-SetFilter angewendet. 3 Die Anzeige wird aktualisiert mit UC-API-Draw.
Alternativer Ablauf		

28

Rücksetzen eines Filters		
ID	UC-GUI-ClearFilter	
Status	In Planung	
Priorität	Mittel	
Akteur	Benutzer	
Kurzbeschreibung	Der Filter wird zurückgesetzt.	
Vorbedingung	Ein Graph wird angezeigt.	
Nachbedingung	Der Graph wird mit allen Kanten angezeigt.	
	Aktion	Reaktion
Normaler Ablauf	1 Im Filtermenü auf 'Filter zurücksetzen' klicken.	2 Es wird UC-API-ClearFilter angewendet. 3 Die Anzeige wird aktualisiert mit UC-API-Draw.
Alternativer Ablauf		

29

Berechnung des Graphenlayouts		
ID	UC-GUI-Layout	
Status	In Planung	
Priorität	Hoch	
Akteur	Benutzer	
Kurzbeschreibung	Das Layout des Graphen wird, basierend auf den vorliegenden Filtern und ein/ausgklappten Knoten, berechnet.	
Vorbedingung	Ein Graph wird angezeigt.	
Nachbedingung	Das Layout des Graphen ist aktualisiert.	
	Aktion	Reaktion
Normaler Ablauf	1 Klicken auf den Button zur Neuberechnung des Layouts.	2 Ein Fenster mit einem Abbruch-Button öffnet sich. 3 Es wird UC-API-Layout aufgerufen. 4 Das Fenster mit dem Abbruch-Button wird geschlossen. 5 Es wird UC-API-Draw aufgerufen.
Alternativer Ablauf		

30

Abbruch der Berechnung des Graphenlayouts		
ID	UC-GUI-LayoutAbort	
Status	In Planung	
Priorität	Hoch	
Akteur	Benutzer	
Kurzbeschreibung	Die Berechung des Graphenlayouts wird abgebrochen.	
Vorbedingung	Es findet im Moment eine Berechnung eines Graphenlayouts statt und das Abbruchfenster ist geöffnet.	
Nachbedingung	Es findet keine Berechnung mehr statt und das Abbruchfenster ist geschlossen.	
	Aktion	Reaktion
Normaler Ablauf	1 Abbruch-Button wird gedrückt.	2 Es wird UC-API-LayoutAbort aufgerufen. 3 Es wird UC-API-Draw aufgerufen.
Alternativer Ablauf		

31

Exportieren von Graphen nach SVG		
ID	UC-GUI-Export	
Status	In Planung	
Priorität	Hoch	
Akteur	Benutzer	
Kurzbeschreibung	Das angezeigte Graphenlayout wird in einer SVG Datei gespeichert.	
Vorbedingung	Ein Graph wird angezeigt.	
Nachbedingung	Es existiert eine SVG Datei die dem Graphenlayout entspricht.	
	Aktion	Reaktion
Normaler Ablauf	1 Klicken auf den Export-Button. 3 Eingabe des Dateinamens und bestätigen.	2 Ein Dateiauswahldialog wird geöffnet. 4 Es wird UC-API-Export aufgerufen.
Alternativer Ablauf		

32

Ausführen eines Skripts		
ID	UC-GUI-RunSkript	
Status	In Planung	
Priorität	Hoch	
Akteur	Benutzer	
Kurzbeschreibung		
Vorbedingung	Ein Graph wird angezeigt. Einem Knoten / einer Kante ist ein Skript zugeordnet.	
Nachbedingung	Das Skript wurde erfolgreich ausgeführt.	
	Aktion	Reaktion
Normaler Ablauf	1 Rechtsklick auf Knoten / Kante. 3 Klicken auf gewünschtes Skript	2 Ein Kontextmenü mit der Auflistung der dem Knoten / der Kante zugeordneten Skripte öffnet sich. 4 Es wird UC-API-RunScript aufgerufen.
Alternativer Ablauf		

33

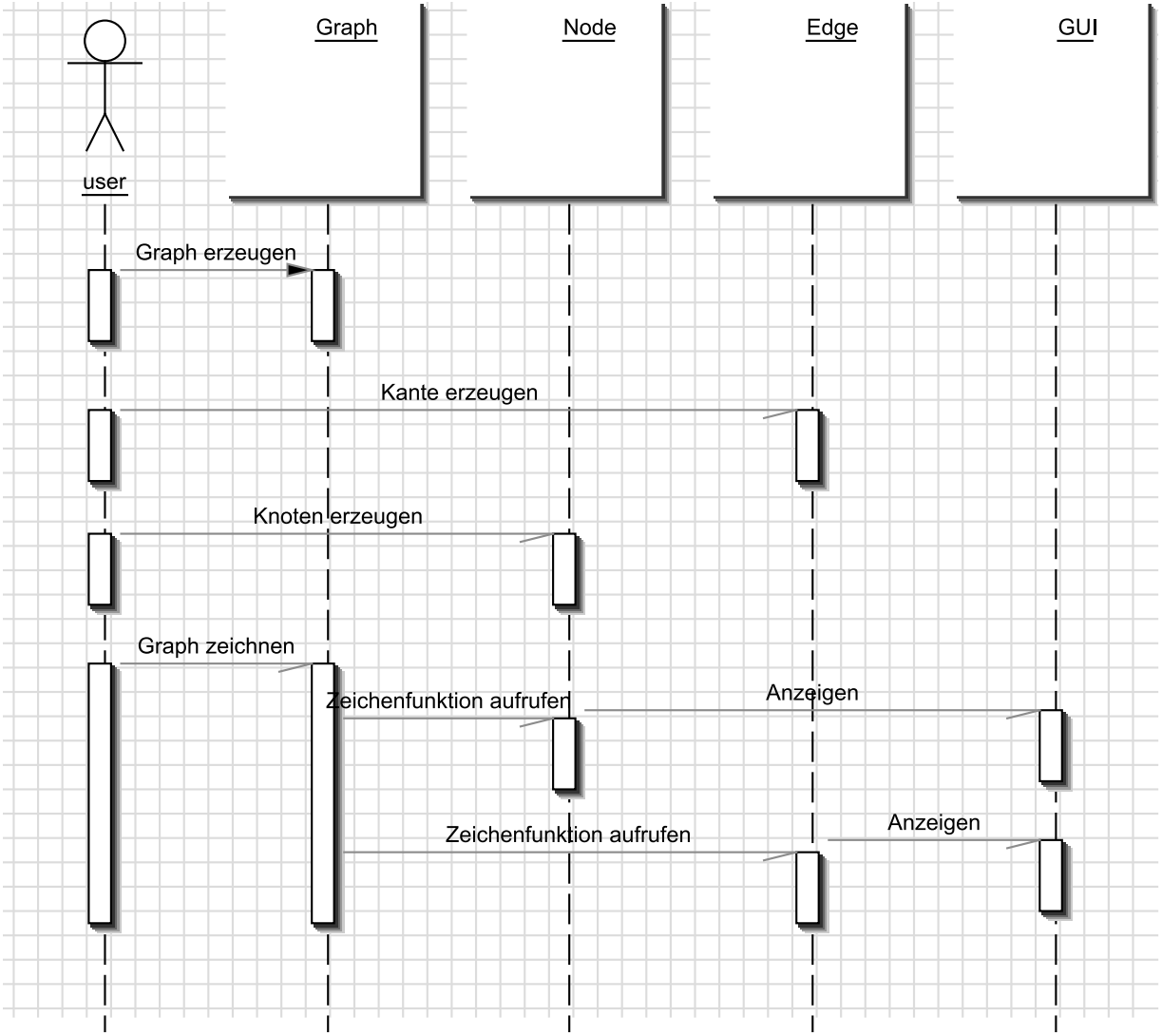
Speichern eines Graphen		
ID	UC-GUI-Save	
Status	In Planung	
Priorität	Hoch	
Akteur	Benutzer	
Kurzbeschreibung	Der Graph inklusive der Skripte, allerdings ohne Layout, wird gespeichert.	
Vorbedingung	Ein Graph wird angezeigt.	
Nachbedingung	Der Graph inklusive der Skripte, allerdings ohne Layout, ist in die Datei geschrieben worden.	
	Aktion	Reaktion
Normaler Ablauf	1 Klicken auf den Speichern-Button. 3 Eingabe des Dateinamens und bestätigen.	2 Ein Dateiauswahldialog wird geöffnet. 4 Es wird UC-API-Save aufgerufen.
Alternativer Ablauf		

34

Laden eines Graphen		
ID	UC-GUI-Load	
Status	In Planung	
Priorität	Hoch	
Akteur	Benutzer	
Kurzbeschreibung	Der Graph wird aus einer Datei geladen.	
Vorbedingung	Installiertes Plugin. Eine Datei im passenden Format zum laden ist vorhanden.	
Nachbedingung	Der Graph wird angezeigt.	
	Aktion	Reaktion
Normaler Ablauf	1 Klicken auf den Laden-Button. 3 Eingabe des Dateinamens und bestätigen.	2 Ein Dateiauswahldialog wird geöffnet. 4 Es wird UC-API-Load aufgerufen. 5 Es wird UC-API-Layout aufgerufen. 6 Es wird UC-API-Draw aufgerufen.
Alternativer Ablauf		

1.5.4. Diagramme

Abbildung 1: Der zum Anzeigen eines Graphens nötige Ablauf



1.6. Qualitätssicherung

1.6.1. Modell

Als grundlegendes Vorgehensmodell verwenden wir das (RUP)-Modell. Das heißt wir verwenden eine iterative Entwicklung die auf komponentenbasierter Architektur aufbaut. Das gewährleistet eine konstante Qualitätssicherung während der Entwicklung. Die einzelnen Disziplinen, des RUP-Modells, stellen die wichtigsten Stationen da, welche das Projekt während der Entwicklung durchläuft. Beginnend bei der Analysephase bis zum fertigen Produkt, kann man dank der Disziplinen und Iterationen bereits früzeitig anfangen zu testenn.

Der Code selbst wird mit Hilfe des Pair-Programmings entwickelt. Dadurch können wir problematische Lösungen vermeiden und verbreiten das Wissen des Quellcodes unter uns, was die Qualität des Endproduktes verbessert.

An technischen Mittel zur Qualitätssicherung verwenden wir SVN als Versionsverwaltungstool. Zudem wird bei einer Code-Änderung eine Email an die Team Mailingliste verschickt, und den Code automatisch auf dem Server kompiliert. Die kompilierten Dokumente werden automatisch in den Downloadbereich und das kompilierte Plugin in die Updatesite der Projekthomepage gestellt. So hat jeder immer die aktuelle, lauffähige Version. So besteht immer die Möglichkeit mit einer realen Installation über die Updatesite den aktuellen Codestand zu testen.
Javadoc und Code Conventions dienen dazu, dass der Code auch leicht durch andere Personen, außer dem Ersteller, lesbar ist. Die Code Conventions sind zB:

- Verwendung von Eclipse Standard-Code-Style
- Methodennamen klein, jedes weitere Wort groß
- kurze und aussagekräftige Variablennamen
- Definitionen (von Variablen, Konstanten) immer am Anfang
- Variablen/Methodennamen auf englisch, Kommentare auf deutsch

Während jeder Iteration unseres Programmes, werden immer wieder Tests durchgeführt. Dazu verwenden wir das JUnit Test-Framework , welches uns Hinweise auf die Art des Fehlers liefert (Falsches Ergebnis, auftreten eines Fehlers). Dadurch können wir garantieren, dass nichts implementiert wird, was nicht fehlerfrei ist. Fehler die Während unseren Erprobungen auftreten dokumentieren wir mit Mantis. Hiermit wird sichergestellt das ein Bug auch dann nicht in Vergessenheit gerät, wenn es nicht direkt eine Möglichkeit gibt, bzw. genügend Zeit zur Verfügung steht, ihn zu beheben.
Sobald es uns möglich ist, werden wir auch mehrere Systemtests durchführen um zu garantieren, dass die einzelnen Komponenten fehlerfrei interagieren.

A. Änderungshistorie

Datum	Thema	Inhalt	seite
05.12.2006	alles	Beginn der History	*

Glossar

Eclipse

Eclipse ist eine offene Entwicklungsplatform, die auf Java-Technologie beruht. *22*

JUnit Test-Framework

JUnit ist ein Framework zum Testen von Java-Programmen, das besonders für automatisierte Unit-Tests einzelner Units geeignet ist *22*

Mantis

Mantis ist ein quellenoffenes, auf MySQL und PHP basierendes Bug-Trackingsystem. Es gibt uns die Möglichkeit einen Überblick über alle noch zu behebenden Probleme zu bekommen. *22*

Pair-Programming

Paarprogrammierung bedeutet, dass bei der Erstellung des Quellcodes jeweils zwei Programmierer an einem Rechner arbeiten. Dabei ist einer für die Erstellung des Codes zuständig, der andere dient zum Kontrollieren *22*

RUP

Rational Unified Process *22*

SVN

Subversion; Team Code Repository *22*

Literatur

[1] http://www-128.ibm.com/developerworks/opensource/library/os-ecplug/.

[2] Eclipse 3.2 Documentation. http://help.eclipse.org/help32/index.jsp.

[3] PDE Does Plug-ins. http://www.eclipse.org/articles/Article-PDE-does-plugins/PDE-intro.html.

[4] SVG Eclipse Plugin. http://sourceforge.net/projects/svgplugin/.

[5] Eclipse.org home. http://www.eclipse.org/, 2006.

[6] Staruml. http://staruml.sourceforge.net/, 2006.

[7] subversion. http://subversion.tigris.org/, 2006.