Bachelor - Praktikum (Graphenlayout)

Pflichtenheft Rev. 2

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

postfuse

one step beyond

Stand: 19. April 2007

`http://bp.macrolab.de`

Tutor: Thorsten Volland
Aufraggeber: Michael Eichberg
Gruppenmitglieder: bp@macrolab.de

- Bastian Christoph bastian.christoph@gmx.de
- Peter Schauss peter.schauss@web.de
- Martin Konrad mkon@gmx.de
- Marco Möller marco.moeller@macrolab.de

Inhaltsverzeichnis

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, welches Ziel mit der Entwicklung des Produktes erreicht werden soll.

Ist gibt Aufschluss darüber, welche Software bisher verfügbar ist. Deren Vor- und Nachteile werden diskutiert, um den Grund für die Entwicklung des neuen Produktes deutlich zu machen.

Soll entspricht der Ziel-Vision im Detail. Alle Features werden einzeln aufgeführt und erläutert, um einen Überblick über das Ziel der Entwicklung zu geben. entwickelte Software zum Schluss abdecken wird. Hieraus lassen sich weitere Anforderungen ableiten.

2 Vision

Es soll ein Eclipse-Plugin entstehen, um kleine bis mittlere Graphen (etwa 1 bis 50 Knoten) zu visualisieren. Um dieses und alle weiteren Plugins einfach und schnell verwenden zu können, sollen die Komponenten über eine Eclipse Update-Site verfügbar sein. Alle verwendeten Plug-Ins sind dabei frei verfügbar, so dass keine Probleme mit abhängigen Lizenzen entstehen.

Die Graphen werden über zwei Schnittstellen definiert. Zum einen über eine Java-API, so dass aus anderen Java-Applikationen heraus Graphen erstellt werden können. Zum anderen über XML-Dateien, da XML mittlerweile ein sehr gebräuchliches Standardformat ist und sich somit sehr gut auch zur Speicherung von Graphenstrukturen anbietet.

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. Es wird auch eine Funktion angeboten, um den angezeigten Graph als Bilddatei zu exportieren.

Die Hauptanwendung wird letztendlich die Visualisierung der Graphen in einem Eclipse-Fenster sein. Dabei ist die Berechnung des Graphenlayouts jederzeit für den Benutzer unterbrechbar.

Bei der Betrachtung von mehreren Graphen werden diese überschaubar und übersichtlich angezeigt. Den Knoten können Texte beliebiger Länge zugeordnet werden und sie köennen auf- und zugeklappt werden.

Die Möglichkeit, Subgraphen und Mehrfachkanten zu verwenden, ist vorgesehen. Die bei Bedarf gerichteten Kanten können ebenfalls beschriftet werden und sind zur besseren Unterscheidung farblich markierbar.

Es werden mehrere Knoten- und Kantenpfeiltypen zur Verfügung gestellt. 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 dem jeweiligen Graphen ausgeführt werden. Grundbefehle, auf die die Skripte zugreifen können, sind im Plug-In implementiert. Es lassen sich mehrere verschiedene Skriptsprachen verwenden. Die Graphen werden zusätzlich in einem Overview Fenster angezeigt, das die Navigation vereinfacht, und unterstützen Pan & Zoom- Funktionen.

3 Ist

Bislang existiert in der Arbeitsgruppe nur ein Plug-In zur Visualisierung von Graphen, das einen Export als Grafik erlaubt. Allerdings stürzt es hin- und wieder 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 möglich, Funktionen wie Zoom in der GUI auszuführen. Auch fehlt eine Möglichkeit, Skripte einzubinden.

4 Soll

Das Ziel unseres 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 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 sind variabel. Es lassen 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.

Priorität	Beschreibung
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.

5 Vergleich Ist/Soll

Wesentliche Neuerungen im neuentwickelten Plug-In sind die Installierbarkeit über eine Update-Site, im Gegensatz zur vorher komplizierten Installation von Hand. Außerdem war bei den alten Tools keine Unterstützung für das Anhängen von Skripten an Knoten und Kanten vorhanden. Zudem konnte die Verarbeitung der Daten nur in mehreren Einzelschritten erfolgen - zentrale Aufgabe im neuen Plugin ist es also, die wichtigen Funktionen in einem Plugin zu vereinen.

Zusammenfassend lässt sich sagen, dass die bisherige Lösung zwar schon einen großen Teil der Funktionalität bietet, aber wenig benutzerfreundlich, unzuverlässig und umständlich ist.

6 Andere existierende Systeme

Im Internet gibt es recht viele Systeme, die sich zum Anzeigen von Graphen eignen, viele sind allerdings kommerziell oder nicht besonders stabil bzw. nicht weit entwickelt. Es gibt aber auch einige Frameworks, die einen stabilen Zustand erreicht haben und bereits in vielen Projekten eingesetzt werden. Im Bereich der Java-Tools sind Prefuse ([11]) und das jung-Framework ([7]) zu erwähnen. Während prefuse sich eher auf die graphische Visualisierung spezialisiert, liegt bei jung der Schwerpunkt eher bei Graphenalgorithmen. Was wir allerdings nicht im Internet finden konnten, ist ein verwendbares, freies Plugin zur Visualisierung von Graphen in Eclipse. Auch die Skriptunterstützung als relativ spezielles Feature war bei keinem der gefunden Tools vorhanden.

7 Anwendungsfälle

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

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

Name: Der eindeutige Name des Anwendungsfalls, der einen Hinweis auf die Funktion gibt.

ID: Eine einzigartige ID, die einen Hinweis darauf liefert, in welchem Zusammenhang der Anwendungsfall mit den Anderen steht, und in welchem Fall er eingesetzt wird (zB API).
Format: UC-[INS|API|GUI]-*

Status: Der Fortschritt der Umsetzung (In Planung, In Arbeit, Abgeschlossen)

Priorität: Wie wichtig die Umsetzung ist (Hoch, Mittel, Niedrig)

Schwierigkeit: geschätzter Aufwand zur Umsetzung: (Hoch, Mittel, Niedrig)

Akteur: Akteur, der den Anwendungsfall auslöst. Dies können bei uns folgende sein:

System Administrator Benutzer, der eine Eclipseinstallation verwaltet. Es wird erwartet, dass er sich mit der Installation von Eclipse sowie der Eclipse-Komponenten über Update-Sites auskennt.

GUI User Benutzer, der mit Hilfe von Maus und Tastatur in Eclipse das Plugin verwendet. Ein GUI-Benutzer kennt sich mit der Verwendung der Eclipse-Oberfläche aus und kennt die typischen Designmuster der Oberfläche von Eclipse (Kontextmenüs fast überall, Editoren, Views).

API User Ein Entwickler, der ein Eclipse-Plugin-Programm schreibt, das über die von uns angebotene Schnittstelle auf die von unserem Plugin zu Verfügung gestellte Funktionalität zugreift. Bei einem Entwickler wird davon ausgegangen, dass er bereits Eclipse-Plugins entwickelt hat und sich mit der Struktur von Eclipse-Plugins auskennt.

Kurzbeschreibung: Eine kurze Beschreibung, was der eigentliche Sinn des Anwendungsfalls ist. Dient dazu, dass der Leser nicht alle Details lesen muss um zu verstehen, worum es geht.

Vorbedingung: Die Vorbedingungen geben an, welche Bedingungen erfüllt sein müssen, damit der Benutzer den Anwendungsfall initiieren kann. Sind diese Bedingungen nicht erfüllt, so funktioniert der Anwendungsfall nicht.

Nachbedingung: Die Nachbedingungen geben an, in welchem Zustand sich das System nach der Ausführung befindet.

Normaler Ablauf: Der detaillierte Ablauf des Anwendungsfalls. In Listenform.

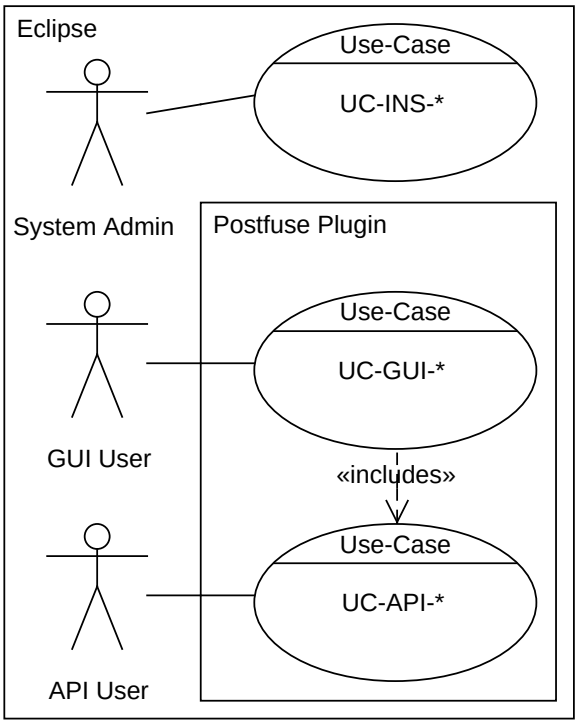
Alternativer Ablauf: Alternative Abläufe. Auch hier wieder als Liste, wobei auf Punkte aus dem normalen Ablauf verwiesen werden kann.

Aktion: Aktive Einflüße die das System beim Durchlaufen des Anwendungsfalls betreffen.

Reaktion: Das direkte Reaktion des Systems auf die unter Aktion ausgeführten Schritte.

In Abbildung 1 lässt sich ein guter Überblick über die Struktur der Anwendungsfälle gewinnen.

Abbildung 1: generisches Anwendungsfalldiagramm



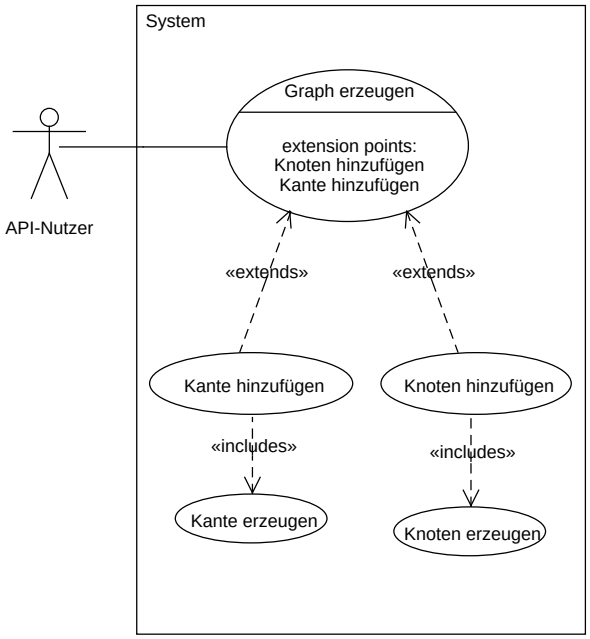
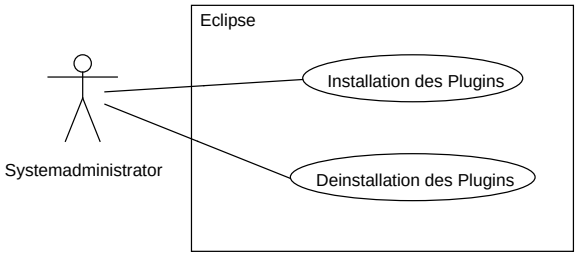


Abbildung 2: Erzeugen eines Graphen über API

7.1 Installations-Use-Cases

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



Installation des Plugins		
ID	UC-INS-Install	
Status	Abgeschlossen	
Priorität	Hoch	
Schwierigkeit	Mittel	
Akteur	Systemadministrator	
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. 3 Der Benutzer startet Eclipse neu.	2 Das Plugin wird mit den auf der Update-Seite angegebenen Parametern installiert.
Alternativer Ablauf		

Deinstallation des Plugins		
ID	UC-INS-DeInstall	
Status	Abgeschlossen	
Priorität	Hoch	
Schwierigkeit	Niedrig	
Akteur	Systemadministrator	
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. 3 Der Benutzer startet Eclipse neu.	2 Das Plugin und alle dazugehörigen Dateien werden gelöscht.
Alternativer Ablauf		

7.2 API-Use-Cases

Ein Teil der Aufgabenstellung fordert, dass unser Programm eine API bereitstellen soll. Die folgenden Anwendungsfälle beschreiben, wie die Interaktion damit 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.

Erzeugen des Graphen		
ID	UC-API-CreateGraph	
Status	Abgeschlossen	
Priorität	Hoch	
Schwierigkeit	Mittel	
Akteur	API User	
Kurzbeschreibung	Erzeugen eines Graph-Objekts	
Vorbedingung	Installiertes Plugin.	
Nachbedingung	Ein Graph-Objekt wurde erstellt.	
	Aktion	Reaktion
Normaler Ablauf	1a Die Methode createGraph() des Plug-ins wird aufgerufen.	2 Das Graph-Objekt wird erzeugt und zurückgegeben.
Alternativer Ablauf	1b Der Graph wird über den Konstruktor erzeugt.	

Hinzufügen eines Knotens		
ID	UC-API-AddNode	
Status	Abgeschlossen	
Priorität	Hoch	
Schwierigkeit	Mittel	
Akteur	API User	
Kurzbeschreibung	Knoten in Graph einfügen	
Vorbedingung	Installiertes Plugin. Es muss ein Graph erzeugt worden sein	
Nachbedingung	Ein Knoten wurde hinzugefügt.	
	Aktion	Reaktion
Normaler Ablauf	1 Die ’addNode’ Funktion des Graph-Objekts, der als Factory dient, wird aufgerufen. Optionaler Parameter ist hierbei ein NodeDesign-Objekt, das die graphische Darstellung definiert.	2 Das Graph-Objekt hat einen Knoten mehr und der erzeugte Knoten wird zurückgegeben.
Alternativer Ablauf		

Hinzufügen eines Subgraphen		
ID	UC-API-AddSubgraph	
Status	Abgeschlossen	
Priorität	Hoch	
Schwierigkeit	Mittel	
Akteur	API User	
Kurzbeschreibung	Subgraphen in Graph einfügen	
Vorbedingung	Installiertes Plugin. Es muss ein Graph erzeugt worden sein	
Nachbedingung	Ein Subgraph wurde hinzugefügt.	
	Aktion	Reaktion
Normaler Ablauf	1 Die 'addSubgraph' Funktion des Graph-Objekts, der als Factory dient, wird aufgerufen. Optionaler Parameter ist hierbei ein NodeDesign-Objekt, das die graphische Darstellung definiert.	2 Das Graph-Objekt hat einen Subgraphen mehr und der erzeugte Subgraph wird zurückgegeben.
Alternativer Ablauf		

Hinzufügen einer Kante		
ID	UC-API-AddEdge	
Status	Abgeschlossen	
Priorität	Hoch	
Schwierigkeit	Mittel	
Akteur	API User	
Kurzbeschreibung	Kante in Graph einfügen	
Vorbedingung	Installiertes Plugin. Es müssen ein Graph- und zwei Kanten-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. Optionaler Parameter ist hierbei ein NodeDesign-Objekt, das die graphische Darstellung definiert.	2a Die Kante wird im Graph-Objekt eingefügt und zurückgegeben.
Alternativer Ablauf		2b Falls ein Fehler bei der Prüfung der beiden Knoten oder eine Graphstrukturverletzung auftritt, wird keine Kante hinzugefügt, sondern eine GraphStructureException geworfen.

Erweitern der Skriptumgebung		
ID	UC-API-ScriptEnv	
Status	Abgeschlossen	
Priorität	Hoch	
Schwierigkeit	Hoch	
Akteur	API User	
Kurzbeschreibung	Es können EInträge zur Skriptumgebung eines Skripts hinzugefügt werden.	
Vorbedingung	Installiertes Plugin.	
Nachbedingung	Die Skriptumgebung wurde um einen Eintrag erweitert.	
	Aktion	Reaktion
Normaler Ablauf	1 Die Skriptumgebung wird vom Skript angefordert. 2 Es wird ein Eintrag hinzugefügt.	3 Die interne Skriptumgebung wird erweitert.
Alternativer Ablauf		

Erzeugen eines Skriptes		
ID	UC-API-CreateScript	
Status	Abgeschlossen	
Priorität	Hoch	
Schwierigkeit	Mittel	
Akteur	API User	
Kurzbeschreibung	Anlegen eines Skripts	
Vorbedingung	Installiertes Plugin.	
Nachbedingung	Es wurde ein Skript angelegt.	
	Aktion	Reaktion
Normaler Ablauf	1 Das Skript wird über die ScriptFactory, die man vom Plugin anfordern kann, erzeugt. Parameter dabei sind: Beschriftung und Code oder Datei sowie der Klassename der Skriptklasse für die gewünschte Skriptsprache.	2 Skript-Objekt wird erzeugt und zurückgegeben.
Alternativer Ablauf		

Binden eines Skriptes an Knoten oder Kante		
ID	UC-API-AddScript	
Status	Abgeschlossen	
Priorität	Hoch	
Schwierigkeit	Mittel	
Akteur	API User	
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 'addScript'-Funktion des Knoten- oder Kanten-Objekts wird aufgerufen.	2 Das Knoten- oder Kanten-Objekt hat ein Skript mehr in seinem Kontextmenü.
Alternativer Ablauf		

Ausführen eines Skripts		
ID	UC-API-RunScript	
Status	Abgeschlossen	
Priorität	Hoch	
Schwierigkeit	Mittel	
Akteur	API User	
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 einer neuen Umgebung aus unter Berücksichtigung der Skriptumgebung.
Alternativer Ablauf		

Setzen eines Filters		
ID	UC-API-SetFilter	
Status	Abgeschlossen	
Priorität	Mittel	
Schwierigkeit	Hoch	
Akteur	API User	
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		

Rücksetzen eines Filters		
ID	UC-API-ClearFilter	
Status	Abgeschlossen	
Priorität	Mittel	
Schwierigkeit	Mittel	
Akteur	API User	
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		

Berechnung des Graphenlayouts		
ID	UC-API-Layout	
Status	Abgeschlossen	
Priorität	Hoch	
Schwierigkeit	Hoch	
Akteur	API User	
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 Layouten des Graphen wird in einem neuen Thread gestartet.	2 Das Layout wird unter berücksichtigung von Filtern und ein- bzw. ausgeklappten Knoten berechnet. 3a Das fertige Layout wird intern gespeichert.
Alternativer Ablauf		3b Bei fehlerhaft definiertem Graphen wird eine Fehlermeldung ausgegeben.

Zeichnen von Graphen in der GUI		
ID	UC-API-Draw	
Status	Abgeschlossen	
Priorität	Hoch	
Schwierigkeit	Mittel	
Akteur	API User	
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 dem angezeigten Graphen ist geöffnet.	
	Aktion	Reaktion
Normaler Ablauf	1 Die Zeichenfunktion wird aufgerufen.	2 Der Graph wird gezeichnet.
Alternativer Ablauf		

Exportieren von Graphen nach SVG		
ID	UC-API-Export-SVG	
Status	Abgeschlossen	
Priorität	Hoch	
Schwierigkeit	Hoch	
Akteur	API User	
Kurzbeschreibung	Das berechnete Graphenlayout wird in einer 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 Die Exportfunktion wird unter Angabe des Dateinamens aufgerufen.	2 Der Graph wird in die entsprechnede Datei exportiert.
Alternativer Ablauf		

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

Speichern eines Graphen		
ID	UC-API-Save	
Status	Abgeschlossen	
Priorität	Hoch	
Schwierigkeit	Mittel	
Akteur	API User	
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		

Laden eines Graphen		
ID	UC-API-Load	
Status	Abgeschlossen	
Priorität	Hoch	
Schwierigkeit	Mittel	
Akteur	API User	
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.	2a Die entsprechenden Objekte werden aus der Datei geladen und angelegt.
Alternativer Ablauf		2b Die Struktur der Datei ist fehlerhaft und es wird eine Exception geworfen.

7.3 GUI-Use-Cases

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

Zoomen der Ansicht		
ID	UC-GUI-Zoom	
Status	Abgeschlossen	
Priorität	Mittel	
Schwierigkeit	Niedrig	
Akteur	GUI User	
Kurzbeschreibung	Die Graphenansicht kann vergrößert und verkleinert werden.	
Vorbedingung	Im Anzeigefenster angezeigter Graph.	
Nachbedingung	vergrößerter bzw. verkleinerter Graph im Anzeigefenster.	
	Aktion	Reaktion
Normaler Ablauf	1a Der Nutzer rollt das Rollrad seiner Maus.	2 Die Größe des Graphens im Fenster ändert sich.
Alternativer Ablauf	1b Der Nutzer klickt auf den Zoomin/out-Button in der Toolbar. 1c Der Nutzer klickt auf Zoomin/out im Popup-Menu, das per Rechtsklick auf den Hintergund des Graphen zu erreichen ist.	

Verschieben der Ansicht (‚Pan‘)		
ID	UC-GUI-Pan	
Status	Abgeschlossen	
Priorität	Mittel	
Schwierigkeit	Niedrig	
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 Graphen im Fenster ändert sich.
Alternativer Ablauf		

Graphgröße an Fenster anpassen		
ID	UC-GUI-ZoomToFit	
Status	Abgeschlossen	
Priorität	Mittel	
Schwierigkeit	Mittel	
Akteur	GUI User	
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	1a Der Nutzer klickt auf den ZoomToFit Button.	2 Der Graph wird vollständig im Fenster angezeigt.
Alternativer Ablauf	1a Der Nutzer benutzt den Eintrag im Kontextmenü des Graphenhintergrunds.	

Anzeigen des Graphen-Overviews		
ID	US-GUI-OverviewShow	
Status	Abgeschlossen	
Priorität	Niedrig	
Schwierigkeit	Hoch	
Akteur	GUI User	
Kurzbeschreibung	Der GUI User 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 GUI-User 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		

Ein- und Ausfalten eines Knoten		
ID	UC-GUI-Fold	
Status	Abgeschlossen	
Priorität	Niedrig	
Schwierigkeit	Niedrig	
Akteur	GUI User	
Kurzbeschreibung	Knoten wird zusammengefaltet, falls er ausgefaltet war bzw. eingefaltet, falls er ausgefaltet war.	
Vorbedingung	Es wird ein Graph angezeigt mit dem Knoten im anderen Faltungszustand.	
Nachbedingung	Der Faltungszustand des Knoten wurde gewechselt.	
	Aktion	Reaktion
Normaler Ablauf	1 Der GUI User klickt auf 'folded' im Kontextmenü des Knoten.	2 Der Knoten wird gefaltet bzw. ausgefaltet. 3 Die Anzeige wird aktualisiert mit UC-API-Draw.
Alternativer Ablauf		

Anzeige zweier Graphen nebeneinander		
ID	UC-GUI-Dual	
Status	Abgeschlossen	
Priorität	Hoch	
Schwierigkeit	Niedrig	
Akteur	GUI User	
Kurzbeschreibung	Es können zwei Graphen nebeneinander angezeigt werden	
Vorbedingung	Es sind zwei Editoren offen, die Graphen zeigen.	
Nachbedingung	Zwei Fenster nebeneinander, in denen jeweils ein Graph zu sehen ist.	
	Aktion	Reaktion
Normaler Ablauf	1 Über die Leiste des Editors wird dieser mit der Maus neben den anderen gezogen.	2 Der Editor wird von Eclipse neu positioniert.
Alternativer Ablauf		

Setzen eines Filters		
ID	UC-GUI-SetFilter	
Status	Abgeschlossen	
Priorität	Mittel	
Schwierigkeit	Mittel	
Akteur	GUI User	
Kurzbeschreibung	Nur gewisse Graphelemente werden angezeigt.	
Vorbedingung	Ein Graph wird angezeigt.	
Nachbedingung	Der Graph wird entsprechend dem Filter gefiltert angezeigt.	
	Aktion	Reaktion
Normaler Ablauf	1 Im Filtermenü werden Filter gesetzt.	2 Es wird UC-API-SetFilter angewendet. 3 Die Anzeige wird aktualisiert mit UC-API-Draw.
Alternativer Ablauf		

Rücksetzen eines Filters		
ID	UC-GUI-ClearFilter	
Status	Abgeschlossen	
Priorität	Mittel	
Schwierigkeit	Niedrig	
Akteur	GUI User	
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 wird auf 'Filter zurücksetzen' geklickt.	2 Es wird UC-API-ClearFilter angewendet. 3 Die Anzeige wird aktualisiert mit UC-API-Draw.
Alternativer Ablauf		

Berechnung des Graphenlayouts		
ID	UC-GUI-Layout	
Status	Abgeschlossen	
Priorität	Hoch	
Schwierigkeit	Hoch	
Akteur	GUI User	
Kurzbeschreibung	Das Layout des Graphen wird, basierend auf dem vorliegenden PreLayoutFilter und ein/ausgklappten Knoten, berechnet.	
Vorbedingung	Ein Graph wird angezeigt.	
Nachbedingung	Das Layout des Graphen ist aktualisiert.	
	Aktion	Reaktion
Normaler Ablauf	1a 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	1a Es wird auf den entsprechenden Eintrag im Kontextmenü, welches durch Rechtsklick auf den Graphenhintergrund erreichbar ist, geklickt.	

Abbruch der Berechnung des Graphenlayouts

ID	UC-GUI-LayoutAbort
Status	In Arbeit
Priorität	Hoch
Schwierigkeit	Hoch
Akteur	GUI User
Kurzbeschreibung	Die Berechung des Graphenlayouts wird abgebrochen.
Vorbedingung	Es findet im Moment eine Berechnung eines Graphenlayouts statt.
Nachbedingung	Es findet keine Berechnung mehr statt.

	Aktion	Reaktion
Normaler Ablauf	1 Es wird auf den Schließen-Button des aktuellen Editors geklickt.	2 Das aktuelle Editor-Fenster wird geschlossen.
Alternativer Ablauf		

Exportieren von Graphen nach SVG

ID	UC-GUI-Export
Status	Abgeschlossen
Priorität	Hoch
Schwierigkeit	Hoch
Akteur	GUI User
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 Es wird auf den Export-Button geklickt. 3 Eingabe des Dateinamens und bestätigen.	2 Ein Dateiauswahldialog wird geöffnet. 4 Es wird UC-API-Export-SVG aufgerufen.
Alternativer Ablauf		

Exportieren von Graphen nach PNG

ID	UC-GUI-Export-PNG
Status	Abgeschlossen
Priorität	Hoch
Schwierigkeit	Hoch
Akteur	GUI User
Kurzbeschreibung	Das angezeigte Graphenlayout wird in einer PNG Datei gespeichert.
Vorbedingung	Ein Graph wird angezeigt.
Nachbedingung	Es existiert eine SVG Datei die dem Graphenlayout entspricht.

	Aktion	Reaktion
Normaler Ablauf	1 Es wird auf den Export-Button geklickt. 3 Der Dateinamen wird eingegeben und bestätigt.	2 Ein Dateiauswahldialog wird geöffnet. 4 Es wird UC-API-Export-PNG aufgerufen.
Alternativer Ablauf		

Ausführen eines Skripts		
ID	UC-GUI-RunSkript	
Status	Abgeschlossen	
Priorität	Hoch	
Schwierigkeit	Mittel	
Akteur	GUI User	
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 Ein Knoten oder eine Kante wird mit dem rechten Mausknopf angeklickt. 3 Das gewünschte Skript wird aus dem Kontextmenü ausgewählt.	2 Ein Kontextmenü mit der Auflistung der dem Knoten / der Kante zugeordneten Skripte öffnet sich. 4 Es wird UC-API-RunScript aufgerufen.
Alternativer Ablauf		

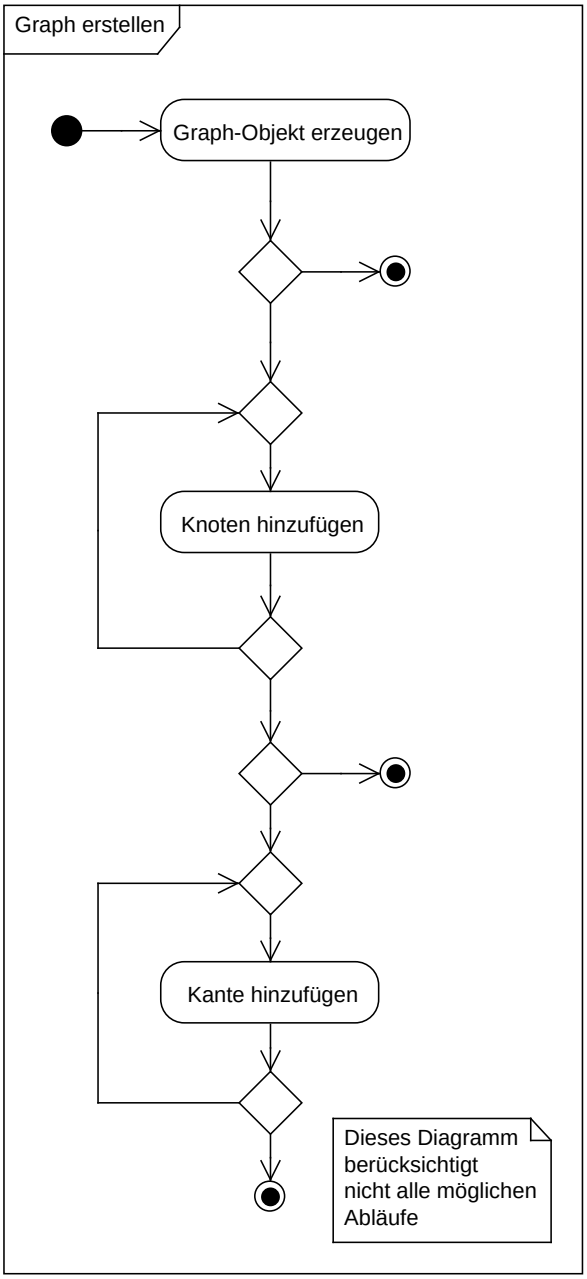
Speichern eines Graphen		
ID	UC-GUI-Save	
Status	Abgeschlossen	
Priorität	Hoch	
Schwierigkeit	Mittel	
Akteur	GUI User	
Kurzbeschreibung	Der Graph inklusive der Skripte, allerdings ohne Layout, wird gespeichert.	
Vorbedingung	Ein Graph wird im aktiven Editor angezeigt.	
Nachbedingung	Der Graph inklusive der Skripte, allerdings ohne Layout, ist in die Datei geschrieben worden.	
	Aktion	Reaktion
Normaler Ablauf	1a Der Speichern-Button von Eclipse wird angeklickt. 3 Der Dateinamens wird eingegeben und bestätigt.	2 Ein Dateiauswahldialog wird geöffnet. 4 Es wird UC-API-Save aufgerufen.
Alternativer Ablauf	1b Klick auf 'Speichern unter' in Eclipse.	

Laden eines Graphen		
ID	UC-GUI-Load	
Status	Abgeschlossen	
Priorität	Hoch	
Schwierigkeit	Mittel	
Akteur	GUI User	
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 Es wird auf den Laden-Button geklickt. 3 Der Dateinamen wird eingegeben und bestätigt.	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		

7.4 Activity Diagramme

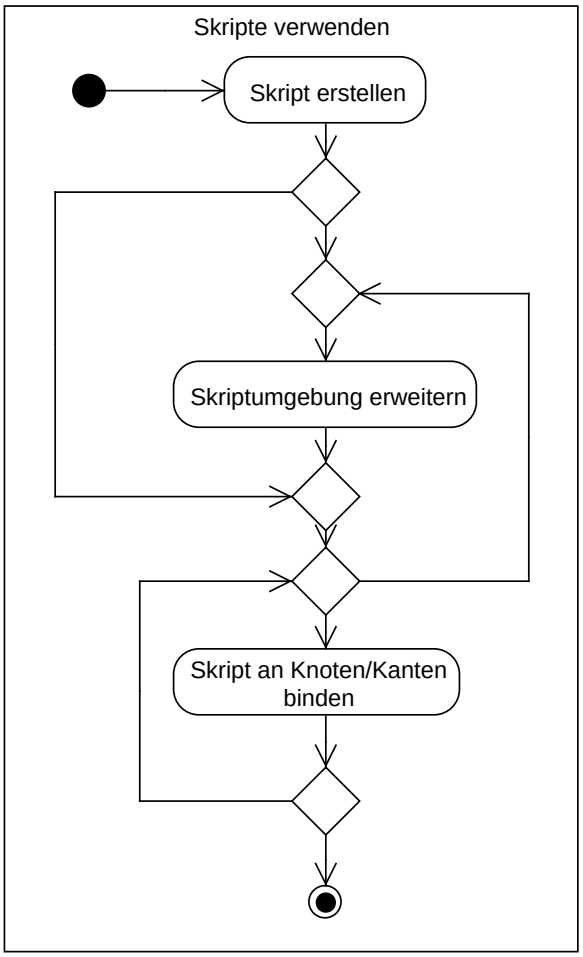
In Abbildung 3 lässt sich ein Überblick über den kompletten Ablauf, der zum Erstellen eines Graphen notwendig ist, gewinnen.

Abbildung 3: Activity Diagramm zum Erstellen eines Graphens

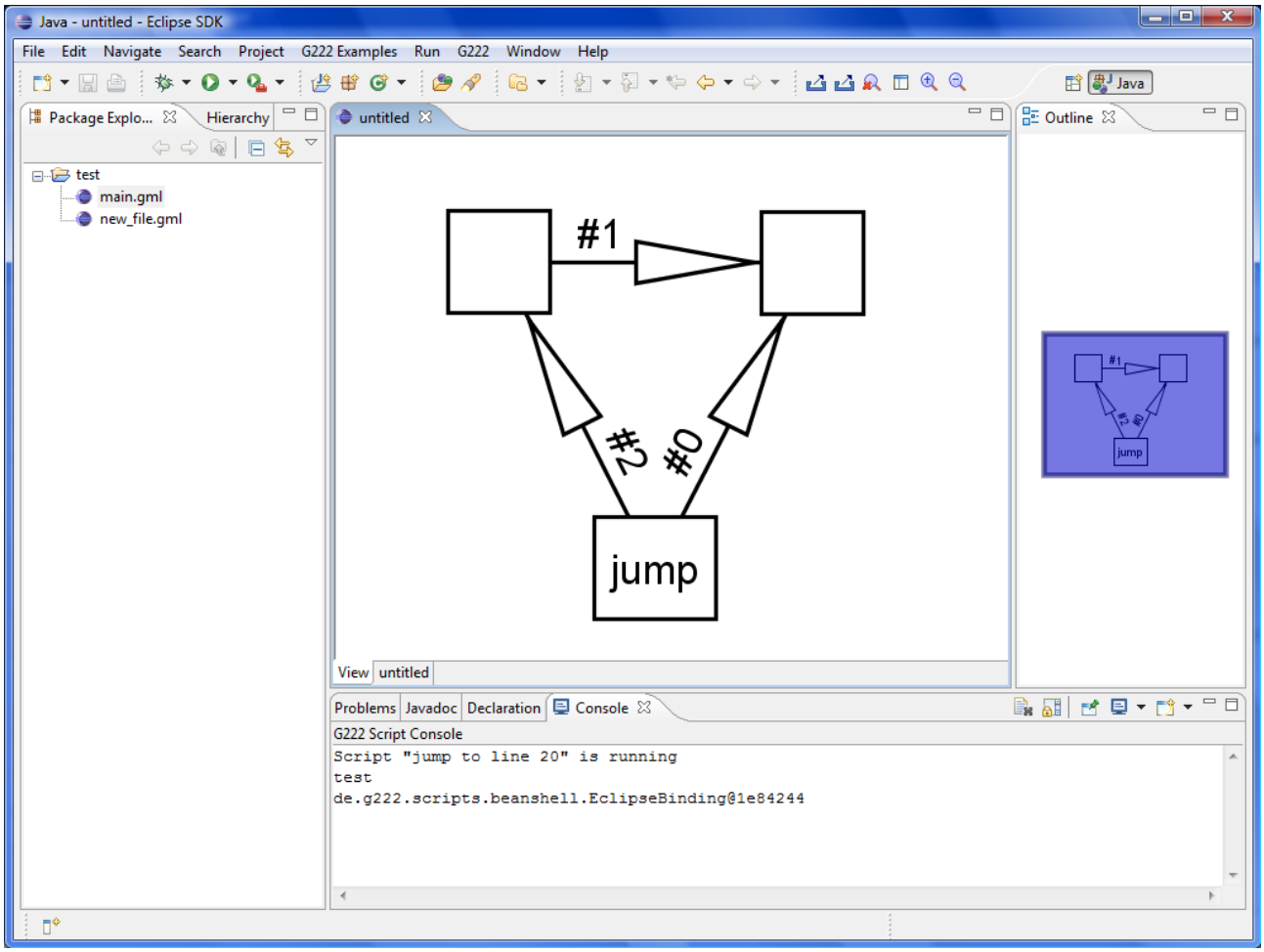


In Abbildung 4 auf der nächsten Seite ist der Ablauf beim Erstellen und Hinzufügen von Skripten dargestellt.

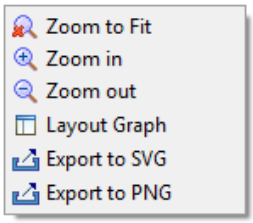
Abbildung 4: Activity Diagramm zum Verwenden von Skripten



8 Benutzeroberfläche



Dieses Bild zeigt die Benutzeroberfläche unseres Plug-Ins. In der Mitte wird der Graph in einem Eclipse-Editor angezeigt. Rechts in der Outline wird eine Übersicht des gesamten Graphen gezeigt, zusammen mit einem Rechteck über dem Bereich, der im Eclipse-Editor angezeigt wird.



Dies ist das Popup-Menü, das erscheint, wenn man mit dem rechten Mausknopf auf einen leeren Bereich im Bild des Graphen klickt.

9 Qualitätssicherung

10 Maßnahmen zur Qualitätssicherung

10.1 RUP

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 dar, welche das Projekt während der Entwicklung durchläuft. Beginnend bei der Analysephase bis zum fertigen Produkt kann man dank der Disziplinen und Iterationen bereits frühzeitig anfangen zu testen.

10.2 Pair-Programming

Der Code selbst wird zum Teil 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.

10.3 Codeverwaltung und -dokumentation

An technischen Mitteln zur Qualitätssicherung verwenden wir SVN als Versionsverwaltungstool. Zudem wird bei einer Code-Änderung eine EMail an die Team-Mailingliste verschickt und der Code automatisch auf dem Server kompiliert. Die kompilierten Dokumente werden automatisch in den Downloadbereich und das kompilierte Plugin auf die Updatesite der Projekthomepage gestellt. So hat jeder immer die aktuelle, lauffähige Version. Es besteht hierdurch 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 für andere Personen als den Ersteller gut lesbar ist. Die Code Conventions sind z.B.:

- 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

10.4 Tests und Bugs

10.4.1 Unit Test

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). Die graphische Oberfläche testen wir von Hand, da eine Automatisierung der Tests den Rahmen des Projekts sprengen würde.

10.4.2 Überdeckung

Damit wir auch sicherstellen können, dass die unsere Tests den gesamten Code Abdecken, messen wir dies mit dem Coverlipse Eclipse-Plugin. Dadurch können wir garantieren, dass nichts implementiert wird, was nicht fehlerfrei ist.

10.4.3 Bugtracking

Fehler, die während unseren Erprobungen auftreten, dokumentieren wir mit Mantis. Hiermit wird sichergestellt, dass ein Bug auch dann nicht in Vergessenheit gerät, wenn es nicht direkt eine Möglichkeit gibt, bzw. genügend Zeit zur Verfügung steht, ihn zu beheben.

10.5 Plattformdiversität

Die Anforderung der System- und Plattformunabhängigkeit wird bei uns immer implizit mitgetestet, da bei uns in der Projektgruppe eine sehr heterogene Hardware- und Systemumgebung anzutreffen ist.

10.6 Nutzen von Modulen

In unserem Projekt wird intensiv von bereits etablierten Komponenten Gebrauch gemacht. Z.B. verwenden wir als Grundgerüst der Graphen Prefuse und für die Skripte entsprechende Interpreter aus dem Umfeld des BSF. Dies stellt sicher, dass es in diesen Bereichen unseres Programms sehr wenige Fehler liegen können, da dieser Code schon in anderen Projekten vielfach genutzt wurde und somit einen gewissen Reifegrad aufweist.

10.7 Abbruchbedingung der Tests

Wir werden die Tests abbrechen, wenn wir keine Fehler mehr finden, die die Stabilität des Plugins gefährden und wir keine Zeit mehr haben, kleinere Probleme in der Oberfläche oder bei Features niedriger Priorität zu beheben. Da die Zeit sowieso für ausführliche Tests recht knapp bemessen ist, werden wir bis zuletzt auf Fehler prüfen.

11 Planung

Um die vielen Abgabezeitpunkte der Dokumentiterationen fristgerecht einzuhalten, haben wir uns eine Zeitplanung überlegt, die in Abbildung 5 zu sehen ist. Die zunächst festgelegte Aufgabenverteilung nach Themen: Marco übernimmt die Einarbeitung in XML-Datenstrukturen. Andreas und Peter evaluieren verschiedene Tools zur Darstellung von Graphen. Martin arbeitet sich in die Eclipse-Plugin-Entwicklung ein. Die Projektleitung übernimmt zunächst Marco, der auch den ersten Review-Vortrag halten wird. Insgesamt ist eine Arbeitszeit von 1000 Mannstunden, d.h. 250 Stunden pro Person von uns für das Projekt veranschlagt.

KW	Datum		Pflichtenheft	Designdokument	Qualitätssicherungs dokument	Implementierung	Test
42	16.10.06						
43	23.10.06						
44	30.10.06		Version 0				
45	06.11.06						
46	13.11.06						
47	20.11.06						
48	27.11.06		Version 1				
49	04.12.06			Version 0			
50	11.12.06		Review				
51	18.12.06						
52	25.12.06						
1	01.01.07						
2	08.01.07			Version 1			
3	15.01.07						
4	22.01.07			Review	Version 0		
5	29.01.07						
6	05.02.07						
7	12.02.07				Version 1	Iteration 1	
8	19.02.07						
9	26.02.07						
10	05.03.07						
11	12.03.07						
12	19.03.07						
13	26.03.07			Überarbeitung	Überarbeitung		
14	02.04.07					Iteration 1.1	
15	09.04.07						
16	16.04.07					Endprodukt	
17	23.04.07						

Abbildung 5: Unsere Zeitplanung

12 Änderungshistorie

Datum	Thema	Inhalt	seite
05.12.2006	alles	Beginn der History mit auslieferung Revison 1	*
03.04.2007	Korrektur	Überarbeitung der Usecases	*
08.04.2007	alles	finales Release 2	*

Glossar

ASCII

Abkürzung für American Standard Code for Information Interchange. Ein weit verbeitete Zeichenkodierung, auf die viele andere aufbauen. `de.wikipedia.org/wiki/Ascii` *4*

BSF

Das Bean Scripting Framework (BSF) ist eine standardisierte Schnittstelle für Interpreter von Skriptsprachen, die eine Interaktion mit Java erlauben. `http://jakarta.apache.org/bsf/` *32*

Coverlipse Eclipse-Plugin

Coverlipse ist eine Erweiterung des JUnit Test-Framework. Es zeigt zu den Tests jeweils an, welche Codezeilen beim Test durchlaufen wurden und kann somit Hinweise auf ungetesteten Code geben. *31*

Eclipse

Eclipse ist eine offene Entwicklungsplatform, die auf Java-Technologie beruht. Es wird sowohl als IDE, als auch für die Entwicklung neuer Programme, verwendet. Mit Plugins lässt es sich beliebig erweitern. *31*

Eclipse Update-Site

Eclipse bietet die Möglichkeit Erweiterungen aus dem Internet nachzuinstallieren. Die unterstützten Update-Sites sind spezielle Webseiten, die dies erlauben. *3*

Eclipse-Plugin

Eine Programm oder Programmpaket, dass sich in der Eclipse-Oberfläche einklinkt und den Funktionsumfang erweitert [13]. *3*

GUI

Eine GUI (Graphical User Interface) erlaubt dem Benutzer eines Programmes, mit diesem zu interagieren. Dies geschieht mit Hilfe von grafischen Elementen wie Fenster, Knöpfe und Listen. *3*

Java

Java ist eine moderne weit verbreitete Programmiersprache.
`de.wikipedia.org/wiki/Java_(Programmiersprache)` *3*

Java-API

Eine (Java)-API, kurz für Aplication Programming Interface, ist eine Schnitstelle mit der ein Programm auf die Funktionen eines anderen zugreifen kann.
`de.wikipedia.org/wiki/Programmierschnittstelle` *3*

JUnit Test-Framework

JUnit ist ein Framework zum Testen von Java-Programmen, das besonders für automatisierte Unit-Tests einzelner Units geeignet ist *31*

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 und alle Featurewünsche zu bekommen. *31*

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, wahrend der andere ständig kontrolliert. *31*

Pan & Zoom

Verschieben, verkleinern und vergößern der aktuellen Bildansicht. *3*

PNG

Portable Network Graphics ist ein im Internet häufig verwendetes verlustbehaftetes Dateiformat für Grafiken. `de.wikipedia.org/wiki/Portable_Network_Graphics` *12*

Prefuse

Ein Java-Framework zum Erstellen und Visualisieren von Graphen, Tabellen und Bäumen. Die Datenfelder lassen sich beliebig mit eigenen Daten erweitern. *6, 32*

RUP

RUP (Rational Unified Process) ist ein Vorgehensmodell für die Entwicklung objektorientierter Programme. Es verwendet UML als Notationssprache. *31*

SVG

Ein auf XML basiertendes 2D-Vektorgrafikformat. Es unterstützt sowohl die üblichen grafischen Primitive, als auch Rastergrafiken, Text und Animationen. *12*

SVN

SVN (Subversion) ist eine Open-Source-Software zur Versionsverwaltung. Es kontrolliert den gemeinsamen Zugriff von mehreren Entwicklern auf die Dateien eines Projektes. *31*

UTF-8

Eine Zeichenkodierung, unter Java der Standard, die eine wesentlich höhere Anzahl von Zeichen abdeckt, als ASCII. `de.wikipedia.org/wiki/Utf8` *4*

XML

eXtensible Markup Language. Eine Gerüsst um Datenformate einheitlich zu spezifizieren. XML beruht auf der Syntax von HTML-Tags
`de.wikipedia.org/wiki/Extensible_Markup_Language` *3*

Literatur

[1] Batik SVG Toolkit. http://xmlgraphics.apache.org/batik/.

[2] Bean Scripting Framework. http://jakarta.apache.org/bsf/.

[3] BeanShell. http://www.beanshell.org/home.html.

[4] Coverlipse. http://coverlipse.sourceforge.net/index.php.

[5] Developing Eclipse plug-ins. http://www-128.ibm.com/developerworks/opensource/library/os-ecplug/.

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

[7] Java Universal Network/Graph Framework. http://jung.sourceforge.net/.

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

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

[10] The GraphML File Format. http://graphml.graphdrawing.org/.

[11] the prefuse visualization toolkit. http://prefuse.org/.

[12] UMLet 7.1. http://www.umlet.com/.

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

[14] StarUML. http://staruml.sourceforge.net/, 2006.

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