Bachelorarbeit am Institut für Informatik der Freien Universität Berlin,

Arbeitsgruppe Software Engineering

Beseitigung von Stolpersteinen im Saros-Entwicklungsprozess

Holger Hans Peter Freyther
Matrikelnummer: 3892143
holger@freyther.de

Betreuer: M.Sc. Franz Zieris
Eingereicht bei: Prof. Dr. Lutz Prechelt

Berlin, 2. Juli 2012

Zusammenfassung

Das Saros-Projekt entwickelt eine Erweiterung für die Eclipse Plattform, um verteilte Paar-Programmierung zu ermöglichen. Die Arbeit an der Erweiterung erfolgt in der Regel im Rahmen von Bachelor-, Master- und Diplomarbeiten. Meine Arbeit hat zum Ziel, durch Änderung des Prozesses, der Architektur und Verbesserung der Dokumentation den Einstieg und die Entwicklung an der Erweiterung zu vereinfachen.

Die Arbeit besteht aus vier Teilen. Zuerst werden bekannte und neue Probleme beschrieben, dann beschäftige ich mich mit der Verbesserung der verwendeten Prozesse, technischen Änderungen und Dokumentation, um das Ziel der Arbeit zu erreichen und zum Schluss gebe ich eine Bewertung meiner Arbeit ab.

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel, wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

2. Juli 2012

Holger Hans Peter Freyther

Inhaltsverzeichnis

1 Einführung

> Jedem Anfang wohnt ein Zauber inne.
>
> Hermann Hesse

1.1 Ziel

Das Ziel meiner Arbeit war es den Einstieg für neue Entwickler und die Entwicklung in Saros zu vereinfachen. Es war mir nicht vorgegeben, wie ich dieses Ziel zu verfolgen habe.

1.2 Ansatz

Ich hatte vor dem Beginn meiner Arbeit weder an, noch mit Saros gearbeitet, daher waren mir eventuelle Probleme des Projektes nicht bekannt und ich musste das Projekt erst einmal kennenlernen. Ich entschied mich dafür, durch das Lesen des Quelltextes ein grobes Gefühl für den Aufbau und die Architektur von Saros zu bekommen. Anschließend konnte ich einige kleine und triviale Änderungen, wie z.B. das Beheben von Rechtschreibfehlern, vornehmen. Ich versuchte, diese Änderungen mit dem existierenden Entwicklungsprozess in den Hauptentwicklungszweig von Saros integriert zu bekommen, da ich zu diesem Zeitpunkt noch keine Schreibrechte für das Projekt hatte, und dabei Probleme mit der Prozessdokumentation und mit dem Prozess selbst zu identifizieren. Zusätzlich las ich einige der aktuellen Saros-Abschlussarbeiten, um meine Erfahrungen mit denen anderer zu vergleichen.

Nach diesen ersten Erfahrungen entschied ich mich, mein Ziel durch die Veränderung des Durchsichtsprozesses, der Verwendung der Mailinglisten, Änderungen an Saros selbst, z.B. Korrekturen an der Architektur und durch Dokumentation zu erreichen.

1.3 Aufbau der Arbeit

Diese Arbeit ist folgendermaßen aufgebaut: In den folgenden Abschnitten dieses Kapitels stelle ich das Saros-Projekt und die Eclipse-Erweiterung weiter vor, in Kapitel 2 gehe ich auf bereits bekannte und neue Probleme ein. In den dann folgenden Kapiteln beschäftige ich mich mit von mir durchgeführten bzw. von mir angeregten Änderungen. In Kapitel 3 beschreibe ich die Prozessänderungen im Saros-Projekt, in Kapitel 4 gehe ich auf technische Änderungen im Saros-Projekt ein und in Kapitel 5 beschreibe ich die erstellte bzw. verbesserte Dokumentation. Zum Schluss bewerte ich meine Arbeit und gebe einen Ausblick.

1.4 Was ist das Saros-Projekt

Das Saros-Projekt entwickelt eine Eclipse-Erweiterung für die Eclipse Plattform mit dem Ziel, verteilte Paar-Programmierung zu ermöglichen. Das zentrale Konzept ist das der Sitzung mit zwei oder mehreren Teilnehmern. In einer Sitzung können die Teilnehmer Projekte teilen, diese werden dann synchronisiert und die Dateien der Projekte können von den verschiedenen Benutzern gleichzeitig editiert werden. Die Eclipse-Erweiterung hält im Laufe einer Sitzung alle Dateien zwischen den Teilnehmern konsistent. Es ist für alle Teilnehmer möglich zu sehen, welche Dateien von anderen Teilnehmern geöffnet waren. Eventuelle Textauswahlen der einzelnen Teilnehmer sind für alle sichtbar und werden farblich hervorgehoben. Die Teilnehmer einer Sitzung können per Chat oder VoIP miteinander kommunizieren. Eine weitere Fähigkeit von Saros ist es, den eigenen Bildschirminhalt anderen Teilnehmern als Videostrom zur Verfügung zu stellen.

Die erste Version von Saros entstand als Diplomarbeit von R. Djemili[1]. Die Weiterentwicklung der Eclipse-Erweiterung wurde bisher fast ausschließlich am Fachbereich der Informatik betrieben, so gab es bis zum 30.04.2012 insgesamt 4109 Änderungen im SVN *Repository* von 40 verschiedenen Benutzerkonten. Von den 40 Benutzerkonten wurde eins im Rahmen der Arbeit eines Post-Doktoranden benutzt, drei von Doktoranden, 16 von Studenten im Rahmen der Diplomarbeit, vier bei Masterarbeiten, 12 bei Bachelorarbeiten und eins bei einer Studienarbeit.

1.5 Wie arbeitet das Projekt

Das Saros-Projekt ist unter dem Namen **dpp** auf Sourceforge[1] zu finden und benutzt dort den *Bugtracker*[2], *Mailinglisten*[3] und das SVN Repository[4]. An der Freien Universität Berlin werden zusätzlich weitere Teile der Projektinfrastruktur betrieben. Dazu zählt eine weitere Mailingliste beim Spline[5], eine Software, um Durchsichten für Quelltextänderungen zu ermöglichen, dem **Reviewboard**[6], ein Werkzeug um Umfragen online durchzuführen, dem **phpESP**[7] und eine Installation eines Systems für *Continous Integration*, dem **Jenkins**[8]. Jenkins wurde von Stefan Rossbach als Teil seiner Bachelorarbeit [2] installiert und wird weiterhin durch ihn betreut.

[1] http://sourceforge.net/projects/dpp
[2] http://sourceforge.net/tracker/?group_id=167540
[3] http://sourceforge.net/mail/?group_id=167540
[4] http://sourceforge.net/scm/?type=svn&group_id=167540
[5] Studentisches Projekt Linux Netzwerk, http://spline.de/
[6] http://saros-build.imp.fu-berlin.de/review
[7] http://saros-build.imp.fu-berlin.de/phpESP
[8] http://saros-build.imp.fu-berlin.de/jenkins

Teilprojekt	Aufgabe
de.fu_berlin.inf.nebula	Eine Bibliothek um das Arbeiten mit Eclipse und SWT zu vereinfachen
de.fu_berlin.inf.dpp	Die Eclipse-Erweiterung für die verteilte Paar-Programmierung
de.fu_berlin.inf.dpp.whiteboard	Eine elektronische Tafel

Tabelle 1: Teilprojekte des Saros-Projektes

Innerhalb des Quelltextes gibt es verschiedene Teilprojekte, eine Übersicht befindet sich in Tabelle 1. Jede Änderung am Quelltext brauchte eine Durchsicht, dazu musste der Student seine Änderung in das Reviewboard stellen und auf die Bewertung der Änderung warten. Nach positiver Bewertung durfte die Änderung in das SVN Repository eingepflegt werden. Auf dem *Continous Integration* System werden die Änderungen gebaut und die **JUnit** Tests des jeweiligen Teilprojektes werden ausgeführt. Am nächsten Morgen werden die so genannten STF Tests[3], GUI Ende-zu-Ende Tests, mit den letzten gebauten Versionen der Teilprojekte ausgeführt. Beim Fehlschlagen des Bauens oder Testens wird eine E-Mail an die Entwickler Mailingliste geschickt.

1.6 Technologie

Im Saros-Projekt werden verschiedene Technologien verwendet oder durch meine Arbeit eingeführt. Diese dienen entweder zur Umsetzung des Entwicklungsprozesses oder sind Java Bibliotheken und wurden im Saros-Quelltext benutzt. Die für meine Arbeit relevanten Technologien stelle ich im Folgendem kurz vor.

1.6.1 PicoContainer

Für das Erstellen von vielen Java Objekten in Saros wird die PicoContainer[9] Bibliothek benutzt. Dazu werden alle Klassen, von denen Exemplare erzeugt werden sollen, in den PicoContainer der `SarosContext` Klasse eingefügt und die Aufgabe der Bibliothek ist es, mögliche Abhängigkeiten zwischen den verschiedenen Klassen automatisch aufzulösen und die Objekte in der nötigen Reihenfolge zu erzeugen. Den PicoContainer kann man dann nach einem Exemplar einer Implementierung einer bestimmten Schnittstelle fragen, dies ist als *Dependency Injection* bekannt.

[9] http://picocontainer.codehaus.org/

1.6.2 XMPP

Das *Extensible Messaging and Pressence Protocol* (**XMPP**)[10] ist ein von der Internet Engineering Task Force (IETF)[11] standardisiertes und XML basiertes Protokoll zur Echtzeitkommunikation.

1.6.3 Smack

Smack[12] ist eine in Java geschriebene Implementierung des XMPPs. Saros benutzt diese Bibliothek für die Kommunikation mit den Sitzungsteilnehmern.

1.6.4 XStream

Zum Austausch von Ereignissen und Daten über die Smack Bibliothek werden in Saros Java Objekte mit Hilfe der **XStream**[13] Bibliothek serialisiert und später wieder deserialisiert.

1.6.5 EasyMock und PowerMock

Zum Entwickeln von Modultests benutzt Saros zwei weitere Bibliotheken. Das Schreiben eines Tests kann es erfordern Teile des Systems durch Attrappen zu ersetzen, um ein bestimmtes Verhalten zu simulieren. Das **EasyMock**[14] Framework hilft beim Erstellen dieser Attrappen. Es ist in der Lage für eine Schnittstelle, über den Umweg einer anonymen Klasse, ein Objekt zu erzeugen. Auf diesem Objekt wird das gewünschte Verhalten beim Test durch Methodenaufrufe definiert und kann dann im Testfall benutzt werden. Das **PowerMock**[15] Framework benutzt **EasyMock** und bietet die Möglichkeit, statische und finale Methoden durch Attrappen zu ersetzen.

1.6.6 SWT und SWTBot

SWT ist das **Standard Widget Toolkit**[16] und wird bei Eclipse für die grafische Oberfläche benutzt. **SWTBot**[17] ist ein Testwerkzeug zum Testen von mit SWT programmierten Oberflächen.

[10] http://xmpp.org/about-xmpp/
[11] http://www.ietf.org/
[12] http://www.igniterealtime.org/projects/smack/
[13] http://xstream.codehaus.org/
[14] http://www.easymock.org/
[15] http://code.google.com/p/powermock/
[16] http://www.eclipse.org/swt/
[17] http://swtbot.com/

1.6.7 Saros-Test-Framework

Das Saros-Test-Framework (STF) wurde von Sandor Szücs während seiner Diplomarbeit [4] entwickelt. Es benutzt den SWTBot und ermöglicht es, die grafische Oberfläche per Java Remote Method Invocation (Java RMI) zu steuern.

1.6.8 Git

Git[18] ist ein Versionsverwaltungs-System. Es wurde von Linus Torvalds zur Verwaltung des Linux Quelltextes entwickelt und wird mittlerweile von Juno Hamano betreut. Im Gegensatz zu älteren Systemen wie SVN und **CVS** hat man innerhalb der Arbeitskopie auch ein komplettes Repository. In diesem lokalen Repository können neue *Branches* und *Commits* angelegt werden. Ein Commit beschreibt den Zustand aller Dateien unter Versionsverwaltung. Bei Git ist man an kein zentrales Repository gebunden, sondern es existieren Mechanismen um zwei Repositories je nach Zugriffsberechtigung miteinander zu synchronisieren.

1.6.9 EGit

EGit[19] ist eine Eclipse-Erweiterung für die Team-Funktionalität und bietet eine grafische Oberfläche für häufig benutzte Git Kommandos.

1.6.10 Gerrit

Gerrit[20] ist ein von Google entwickeltes System für Durchsichten. Es zeichnet sich durch die Spezialisierung auf Git aus. Commits können direkt aus dem lokalen Repository in das Gerrit System geschoben werden und erzeugen automatisch eine Anfrage für eine Durchsicht oder aktualisieren eine vorhandene Durchsicht.

1.6.11 Reviewboard

Reviewboard[21] ist eine Webanwendung und ermöglicht es, Durchsichten für Quelltext zu betreiben. Es wurde im Saros-Projekt verwendet.

1.6.12 Jenkins

Jenkins[22] ist eine Anwendung für die *Continous Integration*. Es hat das Konzept eines *Jobs*, z.B. das Bauen eines Java Projektes. Jobs können durch

[18] http://www.git-scm.com
[19] http://eclipse.org/egit/
[20] https://code.google.com/p/gerrit/
[21] http://www.reviewboard.org/
[22] http://jenkins-ci.org/

Ereignisse, wie einen Commit, angestossen werden. Es wird bei Saros zu diesem Zweck verwendet.

1.6.13 JTourBus

JTourBus[23] ist eine Eclipse-Erweiterung und wurde an unserem Fachbereich entwickelt. Sie soll Entwicklern das Verständnis von Programmen erleichtern, dazu werden unter Verwendung der **JavaDoc** Dokumentationssyntax Haltestellen im Quelltext plaziert. Die Haltestellen gehören zu bestimmten Touren und die Eclipse-Erweiterung ermöglicht es, diesen Touren zu folgen.

1.6.14 CMS

Ein *Content Management System* (**CMS**) ist ein System um Inhalte zu editieren und zu publizieren.

1.6.15 Moose

Moose[24] ist eine in Smalltalk geschriebene Plattform zur Analyse von Quelltext und Daten. Moose bietet ein Modell um Informationen über Programme auszudrücken, z.B. welche Methode ruft welche anderen auf, was für Klassen existieren im Programm, welche Klassen benutzen welche anderen Klassen, welche Variablen existieren in welchem Kontext. Es existieren Werkzeuge für die gängigen Programmiersprachen um diese Informationen aus dem Quelltext zu extrahieren und in Moose zu importieren. Innerhalb von Moose ist es möglich, diese Informationen zu durchsuchen und zu betrachten.

[23] https://www.inf.fu-berlin.de/w/SE/JTourBus
[24] http://www.moosetechnology.org/

2 Probleme und Vorgehen

> Probleme kann man niemals mit
> derselben Denkweise lösen,
> durch die sie entstanden sind.
>
> Albert Einstein

In diesem Kapitel zeige ich Probleme im Saros-Projekt auf und beschreibe meinen Ansatz um diese zu lösen. Zuerst nenne ich Probleme, die bereits in anderen Arbeiten beschrieben wurden. Diese wurden zum Teil auch bereits bearbeitet ohne sie jedoch komplett zu lösen, danach nenne ich weitere Probleme, die bisher noch nicht besprochen wurden.

2.1 Bekannte Probleme

2.1.1 Entscheidungen wurden nicht dokumentiert

Entscheidungen zu Architektur und Sollverhalten im Allgemeinen wurden nicht hinreichend dokumentiert. „In den Archiven der Mailinglisten finden sich viele Informationen zum Projekt, leider ist es eher als großer Haufen zu betrachten, welcher durchsucht werden kann. Wichtige Entscheidungen, welche auf der Mailingliste besprochen wurden, werden anderswo nicht festgehalten, wie z.B. die Umstellung von einer projektzentrierten auf eine sessionzentrierte Architektur. Ähnlich sieht es bei Informationen in den Werkzeugen (Review-Board, Testlink) aus. Es ist da, es kann durchsucht werden, aber es kostet Zeit und es ist nicht immer klar, was noch aktuell ist.“ ([5], S. 31).

Bei meinem Einstieg in das Projekt stolperte ich über einen solchen Fall. Ich fand eine Klasse von der kein Exemplar mehr erzeugt wurde, die Funktion dieser Klasse erschien jedoch sinnvoll. Es war zu klären, ob die Klasse aus Versehen nicht mehr benutzt wurde oder ob es eine gewollte Änderung war. Ich fragte[25] auf der Mailingliste nach, bekam aber keine Antwort. Das Fehlen vom Sollverhalten machte eine Korrektur unmöglich.

2.1.2 Schnittstellen wurden nicht benutzt

In Saros werden oft in den Signaturen der Methoden konkrete Implementierungen und nicht Schnittstellen benutzt. Dies gilt bei der Benutzung von Klassen aus Eclipse, aber auch für Klassen aus Saros selbst. In einigen Fällen existiert nur die Implementierung und keine Schnittstelle. „Dies

[25] http://sourceforge.net/mailarchive/forum.php?thread_name=4F4B6E4E.4090100%40inf.fu-berlin.de&forum_name=dpp-devel

kann zu Problemen führen, wenn z.B. eine alternative Implementierung einer Klasse verwendet werden muss, ohne die aktuelle aus dem System zu entfernen.“ ([5], S. 29). Dieses Problem trat besonders beim Testen mit Attrappen auf. So wurde z.B. sehr oft die `SubMonitor` Implementierung an Stelle der `IProgressMonitor` Schnittstelle benutzt.

2.1.3 Vorgeschlagene Modularisierung wurde nicht umgesetzt

Im Saros-Quelltext werden mittels einer Java-Annotierung Klassen einem bestimmten Modul zugeordnet. Diese Modularisierung hat jedoch keine Auswirkung auf die Sichtbarkeit dieser Klassen, da alle Klassen in den PicoContainer des `SarosContext`s eingetragen werden. So ist es einer Klasse aus dem Modul der grafischen Oberfläche möglich, jede andere Klasse als Abhängigkeit zu haben. Klassen aus unteren Schichten kennen die Existenz der grafischen Oberfläche und müssen Teile ihrer Aufgaben im GUI *Thread* ausführen. Belousow2011 [5] schlug eine Modularisierung vor, diese wurde jedoch bisher nicht umgesetzt.

2.2 Neu erkannte Probleme

2.2.1 Einstieg in das Projekt problematisch

Die Arbeitsgruppe möchte Saros als ein OpenSource Projekt führen und auch externen Entwicklern die Mitarbeit am Projekt ermöglichen. Die Hürden zum Einstieg sollten daher so niedrig wie möglich sein. Der Durchsichtsprozess ist elementar für die Mitarbeit, z.B. um eigene Änderungen dem Projekt beizusteuern, und auf der Webseite sollten sich daher alle Informationen zum Erstellen und Durchführen von Durchsichten befinden, dies war jedoch nicht der Fall. Für den Zugang zum Reviewboard brauchte man ein Benutzerkonto und es wurde nicht beschrieben, von wem dieses Konto unter welchen Bedingungen erstellt wird. Der erfolgreiche Einstieg in das Saros-Projekt funktionierte nur mit Hilfe eines Betreuers der Arbeitsgruppe, da dieser die nötigen Informationen zum Erstellen eines Kontos hatte.

2.2.2 Abhängigkeiten und PicoContainer

Für den PicoContainer müssen alle Abhängigkeiten einer Klasse in den Parametern des Konstruktors genannt werden. Der PicoContainer kann dann die Abhängigkeiten zur Laufzeit auflösen und die Objekte in der nötigen Reihenfolge erzeugen. In einigen Fällen wurde aber statt der richtigen Abhängigkeit nur die `Saros` Klasse als Parameter angegeben und die eigentliche Abhängigkeit wurde dann über einen Methodenaufruf erreicht. In einem anderen Fall gab es eine Abhängigkeit zwischen zwei Klassen, diese wurde nicht über den PicoContainer aufgelöst, sondern durch spezielle Behandlung in der `SarosSessionManager` Klasse.

2.2.3 Infrastruktur nicht gewartet

Das Saros-Projekt hatte am Fachbereich der Informatik der Freien Universität Berlin vier virtuelle Maschinen. Auf diesen Maschinen laufen für Benutzer und Entwickler von Saros relevante Dienste. Zu Beginn meiner Arbeit wurde der XMPP Dienst von der *Firewall* des Fachbereiches gesperrt und das System mit Jenkins war nicht mehr erreichbar. Es war auf den Projektseiten nicht dokumentiert, welche Rechner und Dienste existierten oder wer für diese verantwortlich war.

Das Problem mit der *Firewall* hätte einfach vermieden werden können, da, sobald ein *Port* von der Technik geöffnet wird, bekannt ist, wann dieser *Port* wieder geschlossen wird (siehe Anhang A). Bei Wartung der Infrastruktur sollte es keine geschlossenen *Ports* geben. Das Problem mit der Erreichbarkeit von Jenkins wurde von Stefan Rossbach gelöst. Nach Kontakt mit der Technik, um Zugang zu diesen Systemen zu bekommen, verwies mich eine Betreuerin auf die SarosTech[26] Seite im *Wiki* der Arbeitsgruppe. Die Informationen auf dieser Seite waren zum Teil veraltet. Eine Wartung der Infrastruktur fand nicht statt.

2.2.4 Webseite Technologie fehlerhaft

„Die Saros Webseite hat drei Aufgaben zu erfüllen: Sie präsentiert das Projekt nach außen, sie enthält hilfreiche Informationen für die Verwendung von Saros und sie soll den Entwickler am Saros-Projekt bei seiner Arbeit anleiten bzw. unterstützen.“ [6].

Ich habe die Webseite aus der Entwickler-Rolle benutzt. Zu Beginn meiner Arbeit gab es zahlreiche nicht erreichbare Verweise, Anhänge an Seiten sind nicht herunterladbar und Informationen sind veraltet. Die Webseite hat auf jeder Seite einen Verweis zum Editieren des Inhaltes, dieser führt jedoch nur zurück auf die Hauptseite. Meinen Kommilitonen und mir wurde zu Beginn der Abschlussarbeit nicht mitgeteilt, wie man die Webseite editieren kann.

Das Hauptproblem scheint die verwendete Technologie zu sein. So befindet sich die Webseite im **TWiki** des Fachbereiches und alle Wikiseiten mit dem DPP Präfix werden automatisch auf der Saros-Webseite dargestellt. Der Automatismus funktioniert jedoch nicht für die Anhänge und den Verweis zum Editieren der Seite. Der Inhalt der Seite kann im *Wiki* des Fachbereiches editiert werden, nach jeder Änderung erhält man jedoch eine *Forbidden* Meldung, die Änderung wird jedoch übernommen. Das Arbeiten mit diesem System ist umständlich und unangehm.

[26] https://www.mi.fu-berlin.de/wiki/bin/viewauth/SE/SarosTech

2.2.5 Webseite Inhalt veraltet

Auf einigen Seiten sind UML Diagramme als Bild eingebettet, die dazugehörige Quelldatei ist jedoch nicht auf der Seite verfügbar. Kleine Änderungen wie eine Umbenennung einer Schnittstelle brauchen unverhältnismäßig viel Zeit und erfordern die Erstellung eines komplett neuen Diagrammes. Am 13.8.2010 wurde in SVN *Revision* 2320 die `ISharedProject` Schnittstelle in `ISarosSession` umbenannt, die Diagramme auf der Activities[27] Seite benutzen noch heute den alten Namen. Der Inhalt der Webseite ist auch an anderen Stellen zum Teil veraltet.

2.2.6 Verwaister Quelltext

Beim Einarbeiten ging ich im *Package Explorer* von Eclipse alle Java Pakete durch und habe alle Klassen einmal geöffnet. Für jede Klasse habe ich mir dann die *Typ-* und *Call-Hierachie* angeschaut und damit einen Überblick über die Architektur bekommen. Dabei fiel mir einiges im Quelltext auf. Unter anderem gab es zahlreiche nicht mehr ausgeführte Methoden und erzeugte Klassen. In der Regel wurden die Methoden und Klassen früher benutzt und sind nur durch Änderungen verwaist worden, entweder durch nicht vollständiges Entfernen einer Funktionalität oder bei nicht korrekter Transformation bei Veränderung oder Erweiterung.

Die Kosten für diesen verwaisten Quelltext sind schwer zu beziffern. In zwei Fällen ist dies einfacher. Bei meiner Suche fiel mir der `RevertBufferListener` auf, er wurde von dem Benutzer wojtus als *Feature* hinzugefügt, die Klasse sollte beim Rückgängigmachen von Änderungen helfen. Durch eine Änderung zum Hinzufügen von Projekten im Verlauf einer Sitzung wurde sehr wahrscheinlich aus Versehen die Erzeugung des `RevertBufferListener` entfernt. Neben der Klasse selbst wurden dadurch auch andere Klassen und Methoden nicht mehr benutzt, auf den ersten Blick war dies jedoch nicht offensichtlich. Ein weiteres Beispiel war eine Liste von allen gesendeten Aktivitäten, Einträge wurden nur hinzugefügt und niemals entfernt. Die Methoden zum Benutzen dieser Liste existierten nicht mehr. Dadurch wächst der Speicherverbrauch im Verlauf einer Sitzung.

Im Quelltext kann man nicht einfach sehen, ob Teile der Methode zur Laufzeit noch aufgerufen werden oder nicht, die Architektur kann daher auf Entwickler beim Lesen des Quelltextes komplizierter wirken.

[27] http://www.saros-project.org/Activities

2.2.7 Bruch mit der Architektur

Ein weiteres Problem war der Bruch mit der Architektur, besonders bei der Benutzung von Aktivitäten. Es gibt z.B. die *IActivity* Schnittstelle und die Spezialisierung *IResourceActivity*. Diese Spezialisierung hatte ursprünglich eine weitere Methode. Durch Änderungen im Quelltext wurde diese Methode auch in die Basisschnittstelle hinzugefügt. Die Trennung wurde auch an anderen Stellen aufgeweicht.

2.2.8 Nächtliche STF Testfehler

Die Jenkins Installation führt jeden Morgen die STF Tests aus, diese laufen jedoch nicht ohne Fehler durch. Daher wurde jeden Morgen eine Benachrichtigung über die Testfehler an die Entwicklerliste geschickt. Diese E-Mail beinhaltete lediglich einen Verweis auf die Webseite und nicht die fehlgeschlagenen Tests. Die Benachrichtigungen wurden von den aktiven Entwicklern ignoriert.

In der Regel gab es jeden Morgen 10 Testfehler, echte Testfehler, z.B. durch Änderungen im GUI, gingen daher leicht in den anderen Fehlern unter. Nur bei groben Fehlern in der Initialisierung scheiterten deutlich mehr Tests.

2.2.9 Durchsichts-Prozess fehleranfällig

Die größten Probleme waren jedoch mit dem Prozess der Durchsicht. Zuerst musste man mit Eclipse einen *Patch* erstellen, diesen im Browser ins Reviewboard hochladen, die Saros-Gruppe für den Patch auswählen und die Durchsicht dann öffentlich machen. Nach einer erfolgreichen Durchsicht konnte man entweder die Version aus dem lokalen SVN *Checkout* einpflegen oder den *Patch* herunterladen, ihn anwenden und dann *comitten*. Dieser Prozess war manuell und fehleranfällig. So kann man es vergessen, eine neue Datei hinzuzufügen oder unbeabsichtigt das Zeilenende von Unix auf Windows ändern. Es wurde grundsätzlich nicht sichergestellt, dass die Version aus der Durchsicht ohne Änderung in das SVN Repository eingepflegt wurde. Es gab keine Konventionen, die eine Zuordnung von Änderung im SVN Repository zur Durchsicht im Reviewboard möglich machte. Einer Änderung im SVN Repository war nicht zu entnehmen, ob eine Durchsicht dafür stattfand. Die Commit Meldung in einer SVN Änderung war nicht Teil der Durchsicht und wurde erst beim Einpflegen geschrieben.

Beim ersten Besuch im Reviewboard wurde man auf die **RBTools**[28] verwiesen, um von der Kommandozeile einen *Patch* hochzuladen. Dieses Werkzeug

[28] http://www.reviewboard.org/docs/codebase/dev/getting-started/#rbtools

funktionierte jedoch nicht mit der Installation im Projekt. Entwickler bezogen in der Regel den Saros-Quelltext von Sourceforge, auf dem Reviewboard System kam der Quelltext jedoch von *localhost*, dazu wurde ein *cron job* eingerichtet, um regelmäßig das SVN Repository zu spiegeln. Beim Hochladen eines *Patches* mit den **RBTools** verweigerte das Reviewboard die Annahme des Patches, da es nicht die **UUID** des Repositories verglich, sondern die *URL*.

Ein weiteres Problem mit der Spiegelung des SVN Repositories trat bei schnellen Änderungen auf. Ein *Patch* bezog sich auf eine SVN Revision. Das Hochladen eines Patches konnte scheitern, da die letzte Version noch nicht gespiegelt war.

Das letzte Problem war jedoch nur eine Änderung gleichzeitig machen zu können. Als Entwickler hatte man die Wahl entweder auf die Kommentare aus der Durchsicht zu warten, auf diese einzugehen und eine neue Version hochzuladen oder weiter zu entwickeln. Sollte man sich für das Weiterentwickeln entschieden haben, konnte man dies entweder in der aktuellen Arbeitskopie machen oder man legte eine neue Arbeitskopie an. Neue Arbeitskopien boten sich bei der Entwicklung von unabhängigen Änderungen an. Sollte man dagegen, basierend auf einer Änderung, weitere Änderungen erstellen wollen, entwickelte man in der aktuellen Arbeitskopie weiter. In diesem Fall war das Eingehen auf Kommentare aus der Durchsicht relativ problematisch. So befand sich folgende Anfrage des Autors der Änderung im Reviewboard: „Can you quickly fix that and ship it? I have a complex modified incomingprojectnegotiation.java locally with many other unrelated, unverified changes, so it's hard for me to fix the patch“ [7].

2.3 Vorgehen

Einige dieser Probleme wurden bereits in Master- und Diplomarbeiten bearbeitet und nicht gelöst. Es war unwahrscheinlich mit weniger Zeit mehr erreichen zu können. Ich entschied mich daher, einen inkrementellen Ansatz zu verfolgen. Dazu wählte ich das Problem dessen Lösung die grösste nachhaltige Verbesserung bewirken könnte und arbeitete an diesem bis zu dem Punkt, an dem ein anderes Problem wichtiger erschien.

Das Resultat dieses Ansatzes war es, viele kleine Änderungen vorzunehmen, die den Entwicklern die Arbeit an Saros erleichtern und somit dem Projekt helfen. Ein Nachteil an diesem Ansatz war es, vorher nicht sagen zu können, mit welchen Problemen ich mich wie lange beschäftigen würde.

3 Prozessänderungen

> I wish SVN had stashing.
>
> Andreas Haferburg

Hier stelle ich meine Änderungen in der Arbeitsweise des Projektes dar, zu ihnen zählen die Umstellung auf Git und Veränderungen an den Mailinglisten.

3.1 Git und Gerrit

Andreas Haferburg hatte die SVN Integration in Saros implementiert. In den Commit Meldungen für diese Funktionalität fand ich Hinweise auf eine Unzufriedenheit mit SVN und einen Verweis auf eine Fähigkeit von Git, dem so genannten *stashing*. Ich führte auch die beschriebenen Probleme mit dem Reviewboard auf SVN zurück. Ein Umstieg auf Git könnte diese Probleme lösen. Auf der anderen Seite hat Git den Ruf kompliziert zu sein, unter anderem unterstellte Andrew Morton, ein führender Linux Kernelentwickler, Git wäre ausdrücklich entworfen worden um sich weniger intelligent zu fühlen als man vorher dachte es zu sein [8]. Die Gefahr bei einem Umstieg auf Git war daher mehr Probleme durch die Mächtigkeit des Werkzeuges zu schaffen, als ursprünglich mit dem Reviewboard und SVN existierten. Zum Glück besteht mit EGit eine gute Integration mit Eclipse, eine Verwendung der Kommandozeilen-Werkzeuge ist daher nicht notwendig. Mit Hilfe von EGit ist es möglich, einen SVN ähnlichen Arbeitsablauf zu realisieren und gleichzeitig die bekannten Probleme zu lösen. Erfahrene oder neugierige Entwickler können hingegen die volle Funktionalität von Git kennenlernen und benutzen. EGit bietet von Hause aus eine Option an, um mit Gerrit zu arbeiten, Änderungen können direkt aus Gerrit geholt werden und nach Gerrit geschoben werden, so drängte es sich für die Benutzung bei Durchsichten auf.

Ein Git Commit wird direkt aus dem lokalen Repository in das Repository von Gerrit geschoben, dort wird entweder eine neue Durchsicht eröffnet oder eine bereits existierende aktualisiert. Ein Git Commit beinhaltet neben der Änderung auch Informationen zum Autor, eine textuelle Beschreibung und eine Liste von Vorgängern. Gerrit arbeitet direkt auf Git Commits. Nach einer Durchsicht wird der Git Commit je nach Konfiguration direkt oder nach leichter Änderung der Commit Meldung und den Vorgängern in das Git Repository integriert. Ein Git Commit aus dem Saros Repository befindet sich in Abbildung 1. Aus der Kombination von EGit, Git und Gerrit ergeben sich folgende Vorteile gegenüber SVN und Reviewboard:

- Die Commit Meldung ist Teil der Durchsicht und wird nicht mehr nach der Durchsicht gewählt.

- Eine Änderung zwischen Durchsicht und Commit ist nicht mehr möglich.
- Eine Änderung kann mit EGit direkt aus Eclipse in das Gerrit Repository geschoben werden.
- Durch die Benutzung der *Change-Id* in der Commit Meldung existiert eine Zuordnung von Git Commit zur Durchsicht. Je nach Gerrit Konfiguration werden zusätzlich die Durchseher und ein Verweis auf die Durchsicht zur Commit Meldung hinzugefügt.
- Jeder kann selbständig ein Benutzerkonto erstellen, die Teilnahme am Durchsichtsprozess ist damit jedem offen. Für das Erstellen von Durchsichten muss man manuell in eine Gruppe eingetragen werden. Dieser Vorgang wurde für neue Entwickler[29] und Administratoren[30] dokumentiert.

Git hat gegenüber SVN auch einige Nachteile, diese sind zum Teil auf Entwurfsentscheidungen zurückzuführen. Unter anderem fand ich in einer Diskussion[31] der Webkit Entwickler zum Umstieg von SVN auf Git folgende Punkte:

- Es bleibt möglich eine nicht lineare Versionsgeschichte mit vielen *Merges* zu erzeugen, diese kann eventuell schwer nachvollziehbar sein.
- Versions-Nummern sind nicht fortlaufend nummeriert, die Anzahl der Versionen zwischen zwei Commits kann nur mit einem Werkzeug bestimmt werden.
- SVN ist einfacher zu verstehen, lokal existiert dort nur die Arbeitskopie.

[29] http://www.saros-project.org/Guidelines

[30] http://saros-build.imp.fu-berlin.de/sarosGettingStarted/ch07.html#id450554

[31] https://lists.webkit.org/pipermail/webkit-dev/2012-March/019816.html

commit 7e6ba61b083e3a8d3cf0e8d8e5eff34fd0dc9a11
Author: Holger Hans Peter Freyther <holger@freyther.de>
Date: Mon May 28 10:21:36 2012 +0200

[INTERNAL] Create an interface for targeted activities

Saros has several activities that are sent to a specific user
instead of being broadcasted to all of them. Remove knowledge
of specific actitivies from the ConcurrentDocumentClient and
make the StopActivity implement the new interface.

Change-Id: I1aa7007e5e90aa7e19a27b42cdc36403225984fa
Reviewed-on: http://saros-build.imp.fu-berlin.de/gerrit/106
Tested-by: Jenkins CI
Reviewed-by: Stefan Rossbach <srossbach@arcor.de>
Reviewed-by: Holger Freyther <holger@freyther.de>

Abbildung 1: Git Commit über Gerrit

3.1.1 Einführung von Gerrit

Die Einführung von Git und Gerrit brauchte mehrere Anläufe. Zuerst wollte ich Gerrit zu Testzwecken auf den Rechnern des Saros-Projektes installieren, dazu erstellte ich eine Liste von Anforderungen und schickte sie meinem damaligen Betreuer. Nach etwas über einer Woche erhielt ich eine E-Mail von einem ehemaligen Studenten der mir Zugang zu *saros-build* verschaffen konnte. Meine Fragen zu der Installation wurden jedoch nicht beantwortet und das System selbst wurde auch seit längerer Zeit nicht aktualisiert. Zahlreiche Sicherheitskorrekturen waren noch nicht eingespielt, die Paketdatenbank hatte nicht gelöste Konflikte. Ich entschied mich daher, Git und Gerrit zum Testen auf meinem eigenen Rechner zu betreiben.

Mit einer E-Mail[32] lud ich die Entwickler des Saros-Projektes ein EGit, Git und Gerrit zu testen. Ich hatte die Hoffnung, die Umstellung aus der Mitte des Projektes zu machen, in dem ich die aktiven Entwickler von Git und Gerrit begeistere und die Mehrheit zum Arbeiten mit Git und Gerrit bewege. Für eine Übergangszeit konnte ich Änderungen aus Git und Gerrit in das SVN Repository einpflegen, damit blieben das Git und SVN Repository inhaltlich identisch. Dieser Ansatz gefiel jedoch nicht jedem Betreuer und ich wurde aufgefordert, das System sofort auszuschalten.

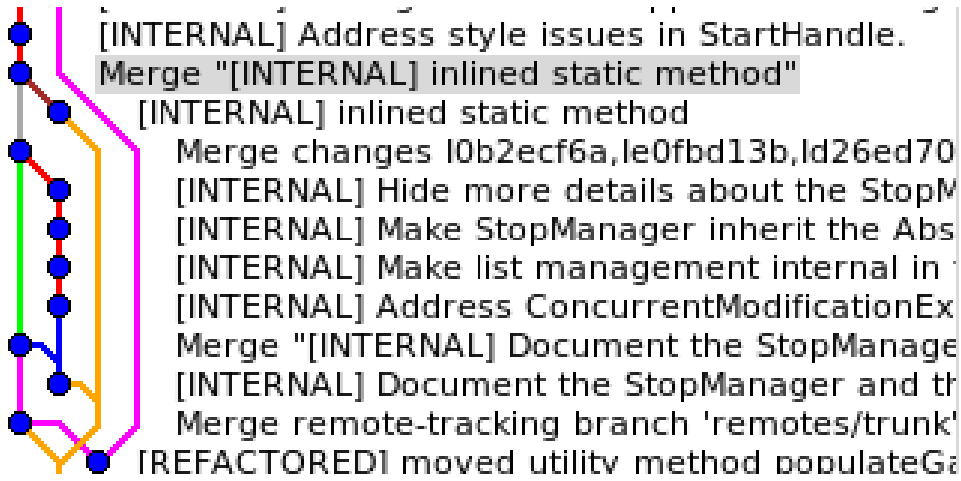
Der nächste Ansatz war die Umstellung anzukündigen. Dazu schickte ich eine weitere E-Mail[33] und schlug die Umstellung für die 17. oder 18. Kalen-

[32] http://sourceforge.net/mailarchive/forum.php?thread_name=4F3FCB23.2070304%40inf.fu-berlin.de&forum_name=dpp-devel
[33] http://sourceforge.net/mailarchive/message.php?msg_id=29090696

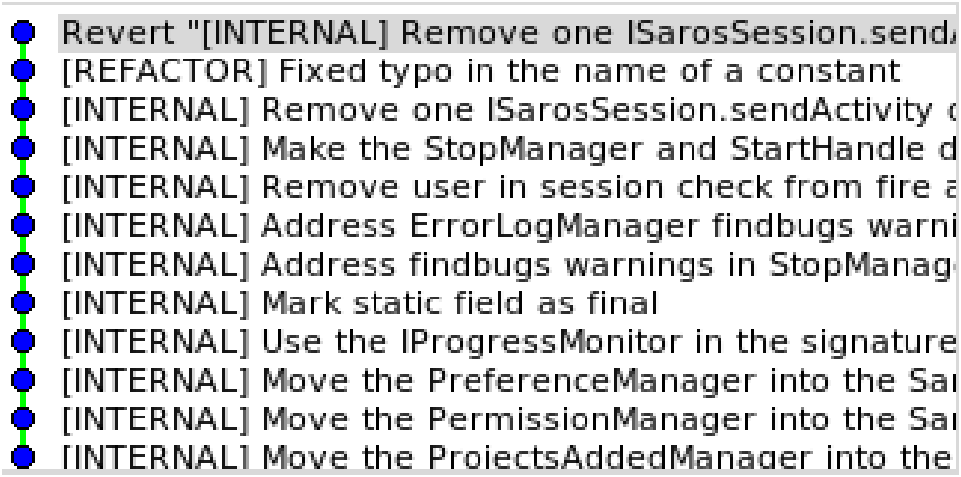
derwoche vor, diese erfolgte dann in der 18. Kalenderwoche.

Gerrit bringt einen Anwendungsserver mit, bei der Installation auf der Saros-Infrastruktur musste ich es jedoch mit einem externen Anwendungsserver betreiben. Daraus resultierten zwei Probleme. Zum einen war es die Aufgabe des Anwendungsservers eine Verbindung zu der Datenbank aufzubauen, dazu brauchte dieser aber Bibliotheken um mit der jeweiligen Datenbank kommunizieren zu können, diese waren aber nicht Bestandteil der Linux Distribution und müssen separat aktualisiert werden. Das zweite Problem war ähnlicher Natur. Eine optionale Bibliothek war nicht im Gerrit Artefakt enthalten. In beiden Fällen musste der Anwendungsserver weitere Bibliotheken laden. Nach einigem Probieren entschied ich mich die `/etc/tomcat6/catalina.properties` zu editieren und das Gerrit `lib` Verzeichnis als `common.loader` einzutragen. Der Vorteil dieser Lösung ist einfacher auf eine neue Version von Gerrit aktualisieren zu können. Der Nachteil ist, andere Anwendungen sind in der Lage die beiden Bibliotheken zu sehen, dies widerspricht der Kapselung von Anwendungen auf einem Anwendungsserver.

Im Gegensatz zu der ersten Installation entschied ich mich bei der endgültigen Installation eine unterschiedliche Konfiguration zum Einpflegen von Änderungen zu wählen. Ursprünglich hatte ich die *Merge if Necessary* Option aktiviert, dies führt zu einer Historie mit vielen *Merges*, das Resultat kann in Abbildung 2(a) gesehen werden. Ein Problem mit dieser Option war das Einpflegen von einer Kette von Änderungen, dort mussten alle vorherigen Änderungen eingepflegt sein, bevor die gewünschte Änderung selbst eingepflegt werden konnte. Dies war beim Betrieb auf meinem eigenen Rechner jedoch ein Problem. Z.B. erstellte ich eine Kette von Änderungen zu einem bestimmten Thema, die aber inhaltlich nicht voneinander abhängig waren. Das Blockieren des Einpflegens wurde dabei eher als störend empfunden. Ich entschied mich daher die *Cherry Pick* Option zu wählen. Mit dieser Option können einzelne Änderungen sofort eingepflegt werden, es werden vor dem Einpflegen weitere Informationen an die Commit Meldung angefügt, z.B. wer diese Änderung durchgesehen hat, bei *Merge if Necessary* ist dies nicht der Fall. Ein Effekt dieser Änderung ist, dass alle Änderungen nur noch einen Vorgänger haben. Die Auswirkung kann in Abbildung 2(b) betrachtet werden.



(a) Merge if Necessary



(b) Cherry-pick

Abbildung 2: Typische Versionsgeschichte bei unterschiedlicher Gerrit Konfiguration

3.2 Mailinglisten

Ich hatte probiert, die öffentliche Entwickler-Liste für meine Fragen und dann später für die Vorstellung meiner Pläne zu benutzen, doch selten bekam ich eine Antwort. Bei den wöchentlichen Treffen am Fachbereich hörte ich dann oft, dass man über diese Entwickler-Liste zu viele E-Mails erhalte und man nicht alle lesen könne. In der Tat gab es zahlreiche E-Mails, z.B. waren es im März 2012 571 Mails, die meisten wurden jedoch von Reviewboard und Jenkins verschickt.

Mein Ziel war es, die Anzahl der zu lesenden E-Mails zu verringern und

somit den Entwicklern die Möglichkeit zu geben, die Entwickler-Liste wieder zu verfolgen. Meine Idee war, eine neue Liste für automatisch erzeugte E-Mails einzuführen. Die Frage war, welche E-Mails auf die neue Liste geschickt werden sollen und was für Auswirkungen dieses haben würde.

Ich betrachtete zuerst die E-Mails vom Reviewboard. Neue Anfragen zu Durchsichten wurden an die Entwickler-Liste geschickt, bei Kommentaren wird der Kommentar an die Entwickler-Liste und alle, die bereits an der Durchsicht beteiligt waren, verschickt. Beim Umleiten der E-Mails auf eine weitere Liste bestand die Gefahr, neue Durchsichten und Kommentare nicht zu sehen. Auf der anderen Seite sollten Entwickler regelmäßig eigene Durchsichten erstellen und somit im Reviewboard selbst neue Anfragen zu Durchsichten sehen. Ich schlug daher vor, die E-Mails vom Reviewboard an die neue Liste zu schicken. Gerrit als Reviewboard Ersatz verschickt keine E-Mails an eine Liste, es ist aber möglich gezielt Personen zur Durchsicht einzuladen, diese erhalten dann Kommentare und Aktualisierungen per E-Mail.

Als nächstes schaute ich mir Jenkins an. In der Regel sollten die E-Mails von Jenkins sofort beachtet werden und eine Reaktion darauf erfolgen. In der Praxis hatten wir jedoch jeden Morgen Testfehler und erhielten somit täglich mindestens eine E-Mail. Der Soll-Zustand ist jedoch, keine Testfehler zu haben und so entschied ich mich, die E-Mails von Jenkins weiterhin an die Entwickler-Liste zu schicken. Ich hatte die Hoffnung, diesen Soll-Zustand noch während meiner Arbeit zu erreichen.

Der Name der neuen Liste ist `dpp-robot` und Franz Zieris setzte meinen Vorschlag um, da ich zu diesem Zeitpunkt noch keine Administrationsrechte für das Projekt bei Sourceforge hatte. Im ersten vollen Monat nach der Umstellung gab es nur noch 83 E-Mails an die Liste, 45 waren davon noch automatisch erzeugt. Einen Rückgang in den Durchsichten habe ich nicht bemerkt.

3.3 Gerrit und Jenkins Integration

Gerrit hat die Möglichkeit für Durchsichten weitere Kriterien hinzuzufügen, diese werden „Kategorien“ genannt. Ein Beispiel für die Darstellung von Kategorien kann in Abbildung 3 gesehen werden. Zusätzlich existiert eine Jenkins-Erweiterung um auf Ereignisse von Gerrit zu warten, dem so genannten Gerrit *Trigger*. Diese beiden Fähigkeiten konnten kombiniert werden, um bei einer neuen Durchsicht oder einer Aktualisierung die Änderung zu bauen und zu testen. Jenkins setzt je nach Ergebnis des *Jobs* entweder eine *+1* oder eine *-1* für die *Verified* Kategorie. In Gerrit können Ände-

rungen, die mit einer *-1* in der *Verified* Kategorie bewertet wurden nicht integriert werden. Mitglieder der *Approvers* Gruppe können dieses Resultat überschreiben, dies war bisher jedoch nicht notwendig.

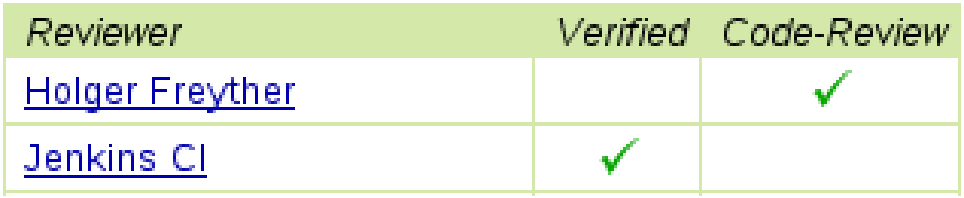


Abbildung 3: Die *Verified* und *Code-Review* Kategorie in Gerrit

Ich erzeugte den Saros-Gerrit *Job* für Jenkins. Dieser baute alle Saros-Teilprojekte und ich entschied mich zuerst, nur die JUnit Testfälle auszuführen. Dies machte ich hauptsächlich aus zwei Gründen:

- Die STF Testfälle liefen nicht stabil, das Testresultat hat nur bei gravierenden Fehlern Aussagekraft.
- Ein Resultat sollte so schnell wie möglich erscheinen. Das Ausführen der JUnit Tests dauerte circa vier Minuten, das Ausführen der STF Tests dauerte hingegen um die 50 Minuten.

Bei der Benuztung von SVN hatte ich ein paar Schwierigkeiten mit Testfehlern nach dem SVN Commit. Diese kamen teilweise durch den Testaufbau, z.B. dem Fehlen einer Verbindung zu dem X11 *Display* oder durch unterschiedliche Versionen der Eclipse-Bibliotheken oder schlicht durch das nicht Ausführen der Tests vor dem SVN Commit. Das Testen bei der Durchsicht und vor dem Einpflegen vermeidet diese Probleme. Auf Grund der *Cherry-Pick* Konfiguration von Gerrit ist es aber weiterhin möglich durch das Missachten einer Abhängigkeit eine nicht bauende Änderung einzupflegen, dies wird dann durch die bereits existierenden Jenkins *Jobs* gefunden.

Das Jenkins System führte den Saros-Gerrit *Job* als `tomcat6` Benutzer aus. Aus diesem Grund musste ich die Konfiguration von Gerrit modifizieren, nur noch Benutzer in einer bestimmten Gruppe dürfen Änderungen per Git hochschieben. Eine Änderung von der Jenkins Konfiguration wäre möglich gewesen, hätte aber zu viel Zeit gebraucht.

3.4 Fazit

Die Umstellung auf Git hat die Probleme mit dem Durchsichtsprozess beseitigt. Es können schneller Durchsichten erstellt werden, Änderungen können

schneller und direkt aus Eclipse heraus getestet werden. Die Integration von Jenkins und Gerrit findet Fehler bevor sie ins Repository gelangen und die Umstellung der Mailinglisten hat die Anzahl der E-Mails deutlich verringert. In Abbildung 4 befindet sich eine Übersicht über das Zusammenspiel vom Saros Git Repository, dem lokalen Repository, und der Interaktion zwischen Gerrit und Jenkins.

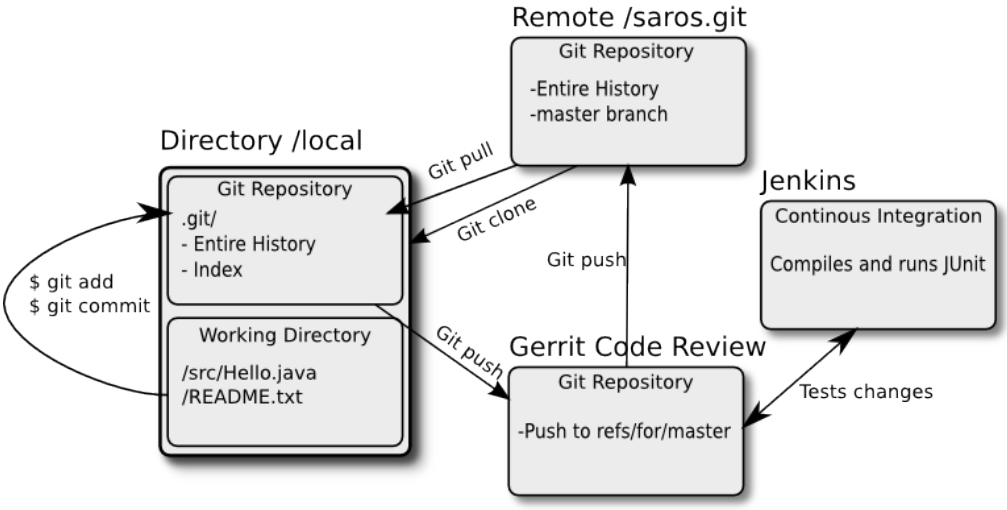


Abbildung 4: Git, Gerrit und Jenkins Übersicht

4 Technische Änderungen

In diesem Kapitel zeige ich meine technischen Änderungen um die vorher beschriebenen Probleme zu lösen. Meine Hoffnung war es durch das Entfernen von Quelltext, Bereinigen der Architektur und dem Schreiben von JUnit Tests die Komplexität von Saros zu verringern, um damit das Arbeiten angenehmer zu machen. Ich erhoffte mir als Seiteneffekt eine Stabilisierung der STF Tests.

4.1 SVN nach Git

Die Umstellung von SVN auf Git und Reviewboard auf Gerrit hatte neben den Änderungen im Prozess auch einen technischen Teil, auf diesen gehe ich an dieser Stelle näher ein. Die fundamentalen Unterschiede zwischen SVN und Git habe ich in Tabelle 2 zusammengefasst, weitere Erklärungen zu Git befinden sich im Anhang dieser Arbeit. Aus den technischen Unterschieden ergaben sich auch andere Konventionen. In einem Git Repository befindet sich in der Regel genau ein Projekt, im SVN Repository von Saros existierten jedoch unabhängige Projekte in verschiedenen Verzeichnissen.

Git bot mit `git-svn` eine Möglichkeit auf einem SVN Repository zu arbeiten, für eine permanente Umstellung von SVN auf Git ist dieses Werkzeug jedoch nicht geeignet, so ist das Aufspalten von Änderungen in verschiedene Repositories und eine spezielle Behandlung von den existierenden Branches nicht möglich. Für die Umstellung von SVN auf Git entschied ich mich daher für das **svn2git**[34] Werkzeug des KDE Projektes. Die Eingabe für dieses Werkzeug ist das Saros SVN Repository, eine Abbildung von SVN Benutzername auf E-Mail Adresse und Regeln zum Aufteilen des SVN Repositories in einzelne Git Repositories. Zuerst sammelte ich ein wenig Erfahrung mit dem Werkzeug und machte eine Testumstellung, dazu machte ich Folgendes:

1. Sourceforge bot das SVN Repository per **rsync**[35] an, mit dem `$ rsync -av dpp.svn.sourceforge.net::svn/dpp/* .` Befehl besorgte ich mir eine Kopie.

2. Mit Hilfe von `$ git log --format=%an | sort -u` erhielt ich eine Liste von verwendeten SVN Benutzernamen. Zuerst probierte ich die Abbildung auf Namen und E-Mail Adressen über die Liste der Abschlussarbeiten herzustellen. Bei einem Benutzer fehlte die Abschlussarbeit, auf Nachfrage erhielt ich die Antwort, dass dieser Benutzer seinen Namen nicht öffentlich genannt haben wollte. An dieser Stelle hatte ich die Wahl entweder alle bisherigen Autoren anzuschreiben und nach zu

[34] https://gitorious.org/svn2git/
[35] http://www.samba.org/ftp/rsync/rsync.html

verwendenden Namen und E-Mail Adressen zu fragen oder die öffentlich zugänglichen Informationen von Sourceforge zu benutzen. Ich entschied mich für den Weg über Sourceforge, da dieser schneller zu realiseren war. Ausserdem blieb es durch das Schreiben einer `.mailmap` Datei möglich später den Namen und die E-Mail-Adresse auf eine andere abzubilden. Die von mir verwendete Abbildung befindet sich im Anhang C, sie benutzt die von Sourceforge öffentlich einsehbaren Informationen.

3. Der letzte Schritt war das Schreiben von Regeln. Dazu fing ich mit einem minimalen Beispiel an. Alle Änderungen ausserhalb des `trunk/` Verzeichnisses führten zur Beendigung des svn2git Werkzeuges und ich konnte eine neue Regel schreiben und danach die Konvertierung fortsetzen. Dies geschah einige Male. Ich entschied mich Saros, Nebula und das Whiteboard in einem Git Repository zu haben, da es Abhängigkeiten zwischen diesen Teilprojekten gab, alle anderen Projekte wurden in andere Git Repositories abgespalten. Die endgültigen Regeln befinden sich im Anhang D.

Die Umwandlung von unseren Branches erforderte jedoch Handarbeit und eine Änderung an svn2git. Der Saros-Veröffentlichungsprozess sah vor, Teile aus dem `trunk/` in verschiedene Branches zu kopieren, z.B. befinden sich Nebula und Saros in verschiedenen Branches. In Git beinhaltet ein Commit aber den Zustand aller Dateien. Nach einer Git Migration würde das Erstellen der Branch für Saros das Löschen aller nicht in die Branch kopierten Dateien bedeuten. Dies erschien mir nicht gewünscht und ich probierte svn2git zu ändern. Beim Erstellen der Branches sollten die nicht kopierten Dateien nicht gelöscht werden, sondern unverändert in der Branch bleiben. Durch das Schreiben von Regelen für svn2git probierte ich die verschiedenen Branches einer Veröffentlichung wieder zusammenzuführen.

Svn2git arbeitet in dem es die Versionshistorie aus SVN extrahiert und in ein textuelles Format bringt, dieses wird vom `git fast-import` eingelesen und es erzeugt daraus die Git Historie. Svn2git verwendete das Schlüsselwort `merge` zum Erstellen einer Branch. Nach dem Lesen der `git fast-import` *manpage* erfuhr ich, dass bei Fehlen eines `from` mit einer Branch ohne Dateien begonnen wird. Dies erklärte das Löschen aller anderen Dateien. Ich machte eine Änderung um beim Erstellen einer *Release Branch* ein `from` zu benutzen. Ich inspizierte einige *Branches* mit einem grafischen Werkzeug und die Versionsgeschichte erschien in Ordnung, das Löschen von nicht *gebranchten* Dateien fand nicht mehr statt.

Nach der endgültigen Migration wollte ich das Resultat überprüfen, auf Grund der Unterschiede zwischen Git und SVN war dies jedoch nicht so

einfach. Änderungen, die nur Verzeichnisse anlegen oder löschen, können in Git nicht ausgedrückt werden, in SVN ist es möglich einen Commit zu haben der Dateien aus einer Branch und dem `trunk` verändert, in Git ist dies nicht in einem Commit möglich. Das Zählen der Commits für jede Branch in Git und SVN und diese zu vergleichen war deshalb nicht sinnvoll. Ich entschied mich aus Zeitgründen für eine reine Plausibilitätsprüfung und schrieb ein Python-Script[36], das den Inhalt aller Branches zwischen SVN und Git verglich. Damit war ich in der Lage zu beweisen, dass der Inhalt am Ende identisch ist, Aussagen über den Weg dorthin konnte ich jedoch nicht treffen. Ich hielt dies für akzeptabel, da man einen Fehler der Versionsgeschichte mit ein wenig Aufwand nachträglich korrigieren könnte.

Die Plausibilitätsprüfung zeigte, dass meine Änderung an svn2git fehlerhaft war. Ich machte meine Änderung unter der Annahme, dass das Erstellen einer Branch durch das Kopieren eines Verzeichnisses entsteht. Es ist jedoch möglich eine Branch lokal vorzubereiten und Dateien aus verschiedenen SVN Versionen zu kopieren. Ich korrigierte meine Änderung in dem ich ein besseres Kriterium für das Erstellen einer Branch fand und fügte dann ein `from` ein. Die Änderung befindet sich im Anhang E und nach dem Wiederholen der Migration war meine Plausibilitätsprüfung erfolgreich und ich veröffentlichte das Resultat bei Sourceforge. Jedem Git Commit wurde über eine Git Notiz die SVN Revision zugeordnet, so bleibt es möglich, zu einem späteren Zeitpunkt eine gründlichere Überprüfung durchzuführen. Eine solche Notiz kann in der Git Commit Meldung in Abbildung 7 gesehen werden.

Typ	SVN	Git
Benutzername	Beliebig	E-Mail Adresse
Versionskontrolle für	Verzeichnisse und Dateien	Dateien
Commit	Änderungen	Zustand aller Dateien
Branch	Kopie eines Verzeichnisses	Verweis auf einen Commit

Tabelle 2: Unterschiede Git und SVN

4.2 Entfernen von Klassen und Methoden

Die in Abschnitt 2.2.6 besprochenen Fehler wurden weder durch unsere automatischen noch durch die manuellen Tests gefunden. Mir erschien es daher sinnvoll systematisch und automatisch nach nicht mehr benutzten Methoden und Klassen zu suchen. Mein Kriterium für eine Methode war, nicht mehr aufgerufen zu werden, bei einer Klasse war es von anderen Klassen nicht mehr benutzt zu werden. Ich suchte zuerst eine Möglichkeit, die Eclipse **Java Development tools** (JDT) mit einem Script zu steuern. Da in der

[36] https://github.com/zecke/saros-svn2git/blob/master/data/verify.py

grafischen Oberfläche die Möglichkeit bestand, sich die Aufrufer einer Methode oder eines Konstruktors zeigen zu lassen hielt ich das für möglich. Die Klassen und Methoden von JDT, die diese Funktionalität implementierten, waren jedoch in einem Paket mit dem Namen **org.eclipse.jdt.internal.ui.callhierarchy** und damit nicht gedacht von anderen benutzt zu werden. Im zweiten Schritt probierte ich Moose zu verwenden. Den Quelltext von Saros importierte ich mit einem Werkzeug namens **Verveine/J**. Beim Importieren gab es ein paar Probleme bei der Importierung von Java Ausnahmen. So trat eine `NullPointerException` auf, wenn eine Ausnahme aus Eclipse im Saros-Quelltext behandelt wurde. Das Werkzeug konnte den Typ nicht in eine Klasse auflösen, da die Eclipse-Bibliotheken nicht eingelesen wurden. Ich konnte das Problem durch das Hinzufügen einer Überprüfung auf `null` vermeiden. In der Zwischenzeit haben die Verveine/J Entwickler das Problem auf eine andere Art gelöst. Nach dem Import musste ich erst einmal Moose kennenlernen, ein Bild der grafischen Oberfläche befindet sich in Abbildung 5. In Moose konnte ich den Saros-Quelltext nach allen Klassen durchsuchen, jede Ansicht erlaubt die Angabe von Smalltalk-Ausdrücken um diese zu filtern, die Eingabe `each clientClasses isEmpty` findet z.B. alle Klassen, die nicht benutzt werden. Diese Liste war grösser als erhofft, z.B. traf dieses Kriterium auf alle Testfälle zu. Mein vollständiges Suchkriterium war daher `(each clientClasses isEmpty and: [(each mooseName findString: 'Test') = 0]) and: [(each mooseName findString: 'anonymous') = 0]`. Damit hatte ich eine Liste von Klassen, die möglicherweise nicht mehr benutzt wurden. Beim Bearbeiten dieser Liste fand ich jedoch zwei Probleme.

- In Eclipse wird die Menüstruktur in XML Dateien beschrieben, in den XML Dateien kann man sich auf Klassen und Methoden beziehen, die zur Laufzeit benutzt werden. Moose war diese Abhängigkeit nicht bekannt.
- Klassen wurden unter Verwendung von `Klasse.class` in den PicoContainer eingefügt. Moose hat den Zugriff auf `.class` nicht als Benutzung von `Klasse` betrachtet.

Meine Liste hatte daher viele falsche Positive und eine automatische Suche nach verwaisten Klassen erschien nur durch Änderung von Verveine/J und Moose möglich. Ich kam zu dem Schluss, dass es gravierendere Probleme in Saros gab und brach meinen Versuch, automatisch nicht mehr benutzte Klassen und Methoden zu finden ab.

Das Finden der ersten nicht benutzten Methoden und Klassen und die darauffolgende systematische Suche nach Klassen resultierte in 19 Änderungen. Die Commit Meldungen befinden sich im Anhang F.

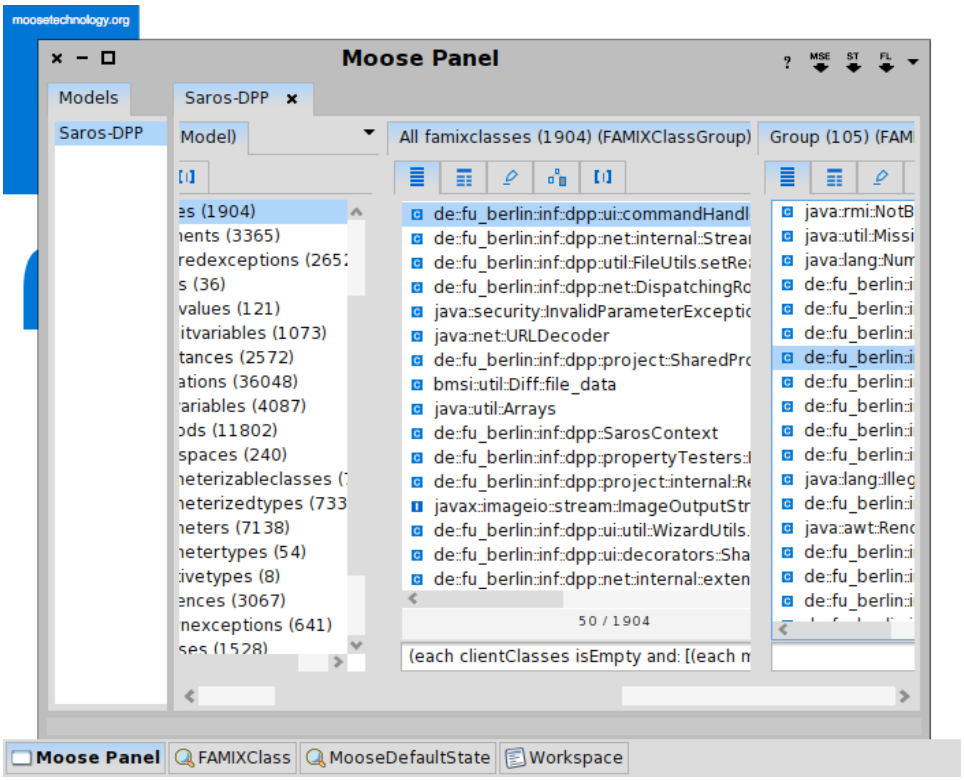


Abbildung 5: Das Finden von Kandidaten mit Moose

4.3 Scoping per Sitzung

In den STF Tests fand ich den `de.fu_berlin.inf.dpp.stf.test.invitation.InviteAndLeaveStressTest` Test, dieser ist nicht Teil der benutzten *Testsuite*. Der Test probiert eine Sitzung aufzubauen und bricht diesen Vorgang nach kurzer Zeit ab. Beim Abbrechen des Sitzungsaufbaus können lokale Ausnahmen signalisiert werden, diese wurden nicht immer behandelt. Das grösste Problem kam aber durch eine Klasse mit dem Namen `StopManager`. Der `StopManager` koordiniert das „Anhalten“ von Nutzern, dazu registrieren sich `Blockable`s beim `StopManager` und diese werden beim Sperren oder Freigeben einer Sitzung benachrichtigt. Zum Anhalten wird ein `StopActivity` Exemplar erzeugt und an die zu sperrenden Nutzer geschickt, diese müssen mit einem weiteren `StopActivity` Exemplar das Sperren bestätigen. Der `StopManager` befand sich innerhalb des PicoContainers des `SarosContexts` und wurde zur Laufzeit genau einmal erzeugt, es war also ein *Singleton*. Das `StopManager` Singleton lauschte auf Änderungen der aktuellen Sitzung und probierte sich dann zurückzusetzen, die Routine kann in Listing 1 gesehen werden.

Der Zustand des `StopManager` Exemplares war an die Dauer einer Sitzung gebunden, die Lebensdauer des Exemplares war aber deutlich länger. Daraus resultierten einige Probleme, unter anderem kann die `sarosSession` Variable nach Beendigung der Sitzung `null` sein, nicht alle Pfade in der Klasse behandelten dies für jeden möglichen Zeitpunkt richtig. Meine Lösung war es den `StopManager` in der `SarosSession` zu erzeugen. Damit konnte ich die `sarosSession` Variable als `final` deklarieren und es konnten keine `NullPointerException` mehr auftreten.

```
protected ValueChangeListener<ISarosSession> sharedProjectObserver = new
    ValueChangeListener<ISarosSession>() {
public void setValue(ISarosSession newSharedProject) {
        if (newSharedProject == sarosSession)
            return;

        // session ended, start all local start handles
        if (newSharedProject == null && sarosSession != null) {
            for (StartHandle startHandle : getStartHandles(sarosSession.
                getLocalUser())) {
                startHandle.start();
            }
            lockProject(false);
        }

        if (sarosSession != null) {
            sarosSession.removeActivityProvider(StopManager.this);
            reset();
        }

        sarosSession = newSharedProject;

        if (newSharedProject != null) {
            newSharedProject.addActivityProvider(StopManager.this);
        }
    }
};
```

Listing 1: StopManager wartet auf Änderung der aktuellen Sitzung

Bei weiterer Suche erkannte ich, dass viele weitere Klassen ihren Zustand an die Sitzung binden, z.B. eine `reset()` Methode haben, aber im PicoContainer des `SarosContexts` als Singleton erzeugt wurden. Die PicoContainer Bibliothek erlaubt es einen Kindbehälter[37] zu erzeugen, so kann ich einen Kindbehälter in der `SarosSession` erzeugen und Klassen, die die Lebensdauer einer Sitzung haben, in diesen Behälter einfügen. Beim Auflösen von Abhängigkeiten wird zuerst der Kindbehälter durchsucht, sollte sich eine Abhängigkeit dort nicht befinden wird bei den Eltern gesucht. Ich erzeugte zuerst den Kindbehälter in der `SarosSession` und verschob den

[37] http://picocontainer.codehaus.org/scopes.html

`StopManager` in diesen und schrieb einen JUnit Test zum Erzeugen, Starten, Anhalten und Verwerfen der Sitzung. Dazu musste ich mit EasyMock Attrappen für alle Abhängigkeiten der `StopManager` und `SarosSession` Klasse schreiben. Danach konnte ich weitere Klassen aus dem globalen `SarosContext` in den Behälter der Sitzung verschieben. Die Umstellung war recht mechanisch und in der Regel machte ich Folgendes:

- Füge ein `implements Startable` ein, um über das Starten und Anhaltens des PicoContainers informiert zu werden.
- Deklariere die `sarosSession` Variable als `final`, da sie sich nicht mehr ändern kann. Für eine neue Sitzung wird ein neues Exemplar benutzt.
- Entferne den `ISarosSessionManager` aus den Parametern des Konstruktors und füge eine `ISarosSession` hinzu, da kein `ISarosSessionListener` mehr installiert werden muss.
- Entferne den `ISarosSessionListener` und benenne die `sessionStarted()` und `sessionStopped()` Methoden in `start()` und `stop()` um, um der `Startable` Schnittstelle zu genügen.
- Entferne die `reset()` Methode und die Aufrufe.

Je nach Klasse musste ich noch weitere Attrappen zu den Tests hinzufügen. Für einige der Attrappen, z.B. dem `EditorManager`, führe ich Buch über die installierten und entfernten *Listener*. Die Implementierung erfolgte mit EasyMock und ist in Listing 2 zu sehen, die Methoden zum Hinzufügen und Entfernen eines *Listeners* fügt das Objekt in einer Liste hinzu oder entfernt es. Am Ende des Tests kann ich dann überprüfen, ob alle *Listener* nach dem Ende einer Sitzung entfernt wurden. In Abbildung 6 befindet sich die Git Commit Meldung als Beispiel für eine Änderung, die durch das Verschieben der Klasse in den PicoContainer der Sitzung das Versagen zur Laufzeit vermeidet.

```
public static EditorManager createEditorManagerMock(
    final List<Object> editorListeners) {
    EditorManager editorManager = EasyMock.createMock(EditorManager.class
        );
    editorManager.addSharedEditorListener(EasyMock
        .isA(ISharedEditorListener.class));
    EasyMock.expectLastCall().andAnswer(new IAnswer<Object>() {

        @Override
        public Object answer() throws Throwable {
            editorListeners.add(EasyMock.getCurrentArguments()[0]);
            return null;
        }
    }).anyTimes();
    editorManager.removeSharedEditorListener(EasyMock
```

```
        .isA(ISharedEditorListener.class));
    EasyMock.expectLastCall().andAnswer(new IAnswer<Object>() {

        @Override
        public Object answer() throws Throwable {
            editorListeners.remove(EasyMock.getCurrentArguments()[0]);
            return null;
        }
    }).anyTimes();
    EasyMock.replay(editorManager);
    return editorManager;
}
```

Listing 2: EasyMock für den EditorManager

commit ecb240408d95f173a56390b0ab9a5d15ad18b7c4
Author: Holger Hans Peter Freyther <holger@freyther.de>
Date: Thu May 10 10:13:43 2012 +0200

[FIX] NPE in the ChangeColorManager during STF Tests

Move the ChangeColorManager into the SarosSession context. This way
sarosSession will always be != null and the NPE can not occur.

java.lang.NullPointerException
at ChangeColorManager$3.userJoined(ChangeColorManager.java:105)
at SharedProjectListenerDispatch.userJoined(SharedProjectListenerDispatch.java:26)
at SarosSession.addUser(SarosSession.java:568)
at OutgoingSessionNegotiation.addUserToSession(OutgoingSessionNegotiation.java:407)
at OutgoingSessionNegotiation.start(OutgoingSessionNegotiation.java:187)
at OutgoingInvitationJob.run(SarosSessionManager.java:466)
at org.eclipse.core.internal.jobs.Worker.run(Worker.java:54)

Change-Id: I2022675b4ce106d9c75a436fa43161b9be382e48
Reviewed-on: http://saros-build.imp.fu-berlin.de/gerrit/66
Tested-by: Jenkins CI
Reviewed-by: Stefan Rossbach <srossbach@arcor.de>

Abbildung 6: Git Commit Meldung zum Verschieben des ChangeColorManagers in den Behälter der `SarosSession`

Beim Verschieben der Klassen zum Sammeln von Statistiken hatte ich ein Problem, am Ende einer Sitzung probiert die Klasse die gesammelten Daten in ein Verzeichnis zu speichern. Der Name des Verzeichnisses, aber auch weitere Namen wurden durch Aufruf von Methoden aus Eclipse geholt, aber Eclipse ist bei meinem Test nicht vollständig initialisert und eine Ausnahme wurde signalisiert. Ich entschied mich daher, den Quelltext zum Speichern der gesammelten Statistiken in eine statische Methode zu verschieben um diese mit PowerMock beim Testen durch eine leere Methode zu ersetzen.

Art	Umgebung
asynchron	Die Ausführung erfolgt in einem neuen Thread.
synchron	Die Ausführung erfolgt sofort.
asynchron SWT	Die Ausführung erfolgt zum nächstmöglichen Zeitpunkt im SWT-Kontext.
synchron SWT	Die Ausführung erfolgt zum nächstmöglichen Zeitpunkt im SWT-Kontext, der Aufrufer blockiert bis die Ausführung beendet wurde.

Tabelle 3: Ausführung bei synchron, asynchron, SWT synchron und SWT asynchron

Insgesamt verschob ich 21 Klassen in den PicoContainer der Sitzung. Klassen der grafischen Oberfläche konnte ich nicht verschieben, da das dynamische Erzeugen von Elementen der grafischen Oberfläche problematisch war.

4.4 Benutzung von SWT im Kern

Ich hatte im Laufe meiner Arbeit einige Attrappen geschrieben und entschied mich diese für das Schreiben eines Tests für den `StopManager` wiederzuverwenden. Dieser Test kann in der `StopManagerTest` Klasse gefunden werden. Dabei stieß ich auf ein allgemeines Problem in Saros. Es existieren mehrere Möglichkeiten ein `Runnable` in einem Kontext auszuführen in dem alle Ausnahmen blockiert werden. Diese Ausführung kann asynchron, synchron, synchron im SWT-Kontext und asynchron im SWT-Kontext geschehen. Eine Übersicht wo und wann die Ausführung stattfindet befindet sich in Tabelle 3.

Die Ausführung im SWT-Kontext ist in der Regel eine Verletzung der Abstraktion. Beim Versuch asynchron SWT im Test zu verwenden gab es Probleme. Der Test wird aus dem SWT-Kontext ausgeführt, beim Warten auf ein Resultat nehme ich der asynchronen SWT Ausführung die Möglichkeit jemals ausgeführt zu werden. Dadurch konnte ich zuerst einige Methoden des `StopManager`s nicht testen. Beim Ausführen der Tests auf dem Jenkins System hatte ich ein weiteres Problem. SWT wird nicht initialisert und jede Verwendung von SWT führt zu der Signalisierung einer Ausnahme.

Um die `StopManager` Klasse testbar zu machen, entschied ich mich den Vertrag der `IActivityListener` Schnittstelle zu ändern. Die `activitCreated` Methode musste vor meiner Änderung aus dem SWT-Kontext aufgerufen werden, nach meiner Änderung war dies weder in der Schnittstelle noch in der Implementierung der Fall. Ich verschob die Ausführung im SWT-Kontext von den Aufrufern zu dem Aufgerufenen, in diesem Fall in die `SarosSession` Klasse. Nach dieser Änderung konnte mein Test auch auf dem Jenkins System erfolgreich ausgeführt werden und die zuvor nicht testbare Methode

wurde testbar.

Die Benutzung von der asynchronen und synchronen Ausführung im SWT-Kontext verhindert das Schreiben von Testfällen. Die Umstellung ermöglichte es mir einen Testfall für die `StopManager` Klasse zu schreiben, aber auch andere `IActivityProvider` wurden durch diese Änderung testbar.

4.5 Activity Architektur

Für das Erstellen und Behandeln von Aktivitäten existieren drei Schnittstellen, `IActivity` (siehe Listing 3) für Aktivitäten selbst, `IActivityProvider` (siehe Listing 4) für Aktivitäten erstellende und bearbeitende Teile und `IActivityListener` (siehe Listing 5), um beim Erstellen von Aktivitäten benachrichtigt zu werden. Das Versenden einer Aktivität über das Netzwerk erfolgt durch eine Implementierung der `IActivityListener` Schnittstelle, dies war z.B. die `SarosSession` Klasse.

```
public interface IActivity {
    public User getSource();
    public void dispatch(IActivityReceiver receiver);
    public IActivityDataObject getActivityDataObject(ISarosSession
        sarosSession);
}
```

Listing 3: IActivity.java

```
public interface IActivityProvider {
    public void exec(IActivity activity);
    public void addActivityListener(IActivityListener listener);
    public void removeActivityListener(IActivityListener listener);
}
```

Listing 4: IActivityProvider.java

```
public interface IActivityListener {
    public void activityCreated(IActivity activityData);
}
```

Listing 5: IActivityListener.java

Diese Architektur wurde jedoch recht schnell gebrochen, denn die `ISarosSession` Schnittstelle war selbst ein `IActivityListener`. Viele Klassen riefen `activityCreated` direkt auf der Sitzung auf und gingen nicht über die installierten *Listener*. Ich entschied mich die Architektur wiederherzustellen, da es neben der `SarosSession` Klasse noch einen weiteren *Listener* gibt und da sich die Architektur anbietet einzelne `IActivityProvider` ohne viel Aufwand zu testen.

Um die Architektur wiederherzustellen, entfernte ich `extends IActivityListener` aus der `ISarosSession` Schnittstelle, die `SarosSession` Klasse hatte fortan eine anonyme Implementierung der `IActivityListener` Schnittstelle, um über die Erstellung von Aktivitäten benachrichtigt zu werden. Ein Aufruf von `activityCreated` ist damit nicht mehr möglich. Es gab weitere Methoden in der `ISarosSession` Schnittstelle um Aktivitäten direkt zu verschicken. Mein Ziel war es, genau einen Weg zu haben, über die Erstellung einer Aktivität zu informieren. Der Unterschied zwischen den Methoden der `ISarosSession` und der `IActivityListener#activityCreated` war, dass die Aktivitäten entweder an eine bestimmte Gruppe von Nutzern geschickt wurden oder an alle. Ich führte die `ITargetedActivity` Schnittstelle als Spezialisierung von `IActivity` ein. Diese Schnittstelle hat eine Methode für die Liste der Empfänger der Aktivität. Um einige der Klassen einfach zu migrieren führte ich die `TargetedActivityWrapper` Klasse als *Adapter* ein. Diese Klasse implementiert die `ITargetedActivity` Schnittstelle und damit konnte ich eine beliebiges `IActivity` Objekt und einen Empfänger zusammenfassen und der `activityCreated` Methode als neues `ITargetedActivity` Objekt übergeben.

4.6 PicoContainer Benutzung

Die Aufgabe des PicoContainer war es Abhängigkeiten aufzulösen, dazu müssen alle Abhängigkeiten der Klasse als Parameter im Konstruktur vorhanden sein, eine Abhängigkeit darf nicht indirekt, z.B. durch einen Funktionsaufruf, aufgelöst werden.

Die falsche Benutzung fiel mir beim Testen mit Attrappen auf. Für einige Abhängigkeiten hatte ich keine Attrappen in meinem Testcontainer und erhielt zur Laufzeit meines Tests eine *NullPointerException* beim Versuch ein `SarosNet` Exemplar zu verwenden. In einem anderen Fall wurde die Auflösung von Abhängigkeiten von Hand umgesetzt, dies fiel mir bei der Umstellung auf einen PicoContainer pro Sitzung auf.

Die Lösung ist einfach. In den einfachen Fällen muss nur die Abhängigkeit als Parameter im Konstruktor hinzugefügt werden, ein Beispiel für diese Korrektur ist in Listing 6 zu finden. In einem anderen Fall konnte ich durch richtige Verwendung des PicoContainers Komplexität aus der `SarosSessionManager` Klasse entfernen, dies ist in Listing 7 zu sehen.

```
-    public SkypeManager(Saros saros) {
-        this.saros = saros;
-        saros.getSarosNet().addListener(this);
+    public SkypeManager(SarosNet sarosNet, IPreferenceStore
    preferenceStore) {
+        this.sarosNet = sarosNet;
+        this.preferenceStore = preferenceStore;
```

```
+         this.sarosNet.addListener(this);

          /**
           * Register for our preference store, so we can be notified if
               the Skype
           * Username changes.
           */
-         IPreferenceStore prefs = saros.getPreferenceStore();
-         prefs.addPropertyChangeListener(new IPropertyChangeListener() {
-             public void propertyChange(PropertyChangeEvent event) {
-                 if (event.getProperty().equals(
-                     PreferenceConstants.SKYPE_USERNAME)) {
-                     publishSkypeIQ(event.getNewValue().toString());
+         preferenceStore
+             .addPropertyChangeListener(new IPropertyChangeListener() {
+                 public void propertyChange(PropertyChangeEvent event) {
+                     if (event.getProperty().equals(
+                         PreferenceConstants.SKYPE_USERNAME)) {
+                         publishSkypeIQ(event.getNewValue().toString());
+                     }
                  }
-             }
-         });
+             });
```

Listing 6: SkypeManager change

```
 public void addSarosSessionListener(ISarosSessionListener listener) {
     if (!this.sarosSessionListeners.contains(listener)) {
-        /*
-         * HACK PreferencesManager relies on the fact that a project is
-         * added only when a session is started, and it might create a
    new
-         * file ".settings/org.eclipse.core.resources.prefs" for the
    project
-         * specific settings. Adding PreferencesManager as the last
    listener
-         * makes sure that the file creation is registered by the
-         * SharedResourcesManager.
-         */
-        if (listener instanceof PreferenceManager) {
-            this.sarosSessionListeners.add(listener);
-        } else {
-        this.sarosSessionListeners.add(0, listener);
-        }
+        this.sarosSessionListeners.add(0, listener);
     }
 }
```

Listing 7: SessionManager dependencies

4.7 Entfernung von IActivityDataObject

In einer Sitzung verschickt Saros Aktivitäten an andere Teilnehmer. Diese Aktivitäten sind vom `IActivity` Typ (Listing 3). In der `SarosSession` Klasse werden diese Aktivitäten in `IActivityDataObject`s (Listing 8) verwandelt und an den `ActivitySequencer` übergeben. Erst beim Versenden in der `XMPPTransmitter` Klasse findet die Serialisierung statt. Die Trennung wurde von Christopher Oezbek im Jahr 2009 eingeführt (siehe Abbildung 7) und der Gedanke schien zu sein den Netzwerkcode von der Anwendungslogik zu trennen. So wird z.B. in der `IActivity` Schnittstelle die `User` Klasse verwendet, in der `IActivityDataObject` Schnittstelle hingegen nur die `JID` Klasse, der XMPP Identität eines Benutzers. `User` Exemplare enthalten Attribute wie z.B. die für Textauswahl zu verwendende Farbe und einen Verweis auf die Sitzung.

Die Umwandlung von `IActivity` zu `IActivityDataObject` erschien mir unnötig. Die `IActivity` Implementierungen sind in der Regel nur Träger von Information und treffen keine Entscheidungen. Das Ausführen von Logik geschieht über eine Implementierung von `IActivityReceiver`. XStream würde es erlauben beim Serialisieren eines `User` nur die `JID` zu serialisieren.

Ich dachte das Erstellen einer neuen `IActivity` Implementierung würde einfacher werden, wenn der Umweg über `IActivityDataObject` nicht stattfindet. Daher probierte ich zuerst die `ISarosSession` Variable aus dem `User` zu entfernen. Der Versuch scheiterte jedoch, da an vielen Stellen der grafischen Oberfläche die Sitzung aus dem `User` Objekt geholt wurde. Bei der Suche nach einem weiteren Ansatz fand ich ein systematisches Problem bei der Deserialisierung in der `SarosSession#hadToBeQueued` Methode (siehe Listing 9). Es ist möglich Aktivitäten zu empfangen, die sich auf ein noch nicht geteiltes Projekt beziehen. Die Umwandlung von `IActivityDataObject` zu `IActivity` scheitert in diesem Fall mit einer Ausnahme. Um dies zu vermeiden werden Exemplare von bestimmten `IActivityDataObject` Implementierungen in eine Warteschlange eingetragen und später verarbeitet. An diesem Punkt entschied ich mich meinen Versuch einzustellen, ich entfernte lediglich nicht mehr benutzte Methoden und Schnittstellen aus den `IActivityDataObject` Implementierungen.

Das obige Problem entstand in der `EditorManager#projectAdded` Methode, diese schickt die Liste der offenen Dateien an alle Teilnehmer der Sitzung bevor das Projekt vollständig geteilt ist. Ein Lösungsansatz wäre es nach dem erfolgreichen Hinzufügen eines Projektes die anderen Teilnehmer nach geöffeneten Dateien zu fragen. Danach könnte die `SarosSession#hadToBeQueued` Methode entfernt werden. Anschliessend ist das Entfernen der `IActivityDataObject` möglich.

```
public interface IActivityDataObject {
    public JID getSource();
    public IActivity getActivity(ISarosSession sarosSession);
}
```

Listing 8: IActivityDataObject.java

commit 6e2101c2de649da706ce1a0a76c720002f239dee
Author: Christopher Oezbek <coezbek@users.sourceforge.net>
Date: Thu Oct 29 14:12:56 2009 +0000

[REFACTOR] Split IActivityDataObjects into IActivityDataObject (for the net component) and IActivity (for the business logic)

At the moment these objects are identical in all but the name!

Conversion is performed in the SharedProject.exec() (at the first point after they are received from the ActivitySequencer) and SharedProject sendActivity() (at the last possible point before they are passed into the ActivitySequencer).

Things still todo:
* The ActivitySequencer (and all net components) should NOT use the User-Object because ait works in a different thread contexti
* Modify the IActivity sub classes to be connected to the shared project properly
* Introduce SPath in the IActivity sub classes
* in the very far future: ConcurrentDocumentManagerServer should have its own SharedProject as it also is running outside the SWT-Thread context.

Notes:
svn path=/trunk/dpp/; revision=1838

Abbildung 7: Commit Meldung Christopher Oezbek

```
    /**
     *
     * @param dataObject
     * @return <code>true</code> if this activity can be executed now
     */
    private boolean hadToBeQueued(IActivityDataObject dataObject) {
        if (!(dataObject instanceof EditorActivityDataObject))
            return false;

        String projectID = ((EditorActivityDataObject) dataObject)
```

```
            .getProjectID();

        /*
         * some activities (e.g. EditorActivity) can return null for
             projectID
         */
        IProject project;
        if (projectID != null) {
            lastID = projectID;
            project = getProject(projectID);
        } else {
            project = getProject(lastID);
        }
        /*
         * If we don't have that shared project, but will have it in
             future we
         * will queue the activity.
         *
         * When the project negotiation is done the method
         * execQueuedActivities() will be executed
         */
        if (project == null) {
            log.info("Activity " + dataObject.toString() + " for Project
                "
                + projectID + " was queued.");
            if (!queuedActivities.containsValue(dataObject)) {
                if (projectID == null) {
                    queuedActivities.put(lastID, dataObject);
                } else {
                    queuedActivities.put(projectID, dataObject);
                }
            }
            return true;
        }
        return false;
    }
```

Listing 9: SarosSession.hadToBeQueued

4.8 Fazit

Die Umstellung von SVN auf Git erforderte eine Anpassung des svn2git Werkzeuges, verlief aber relativ problemlos. Das Suchen und Löschen von verwaistem Quelltext hat Probleme aufgedeckt, eine automatische Suche lies sich leider nicht realisieren. Die Umstellung auf einen Kindbehälter pro Sitzung vermeidet `NullPointerException`s und hat das Potential Versagen auf eine Sitzung zu limitieren. Das Schreiben von Attrappen und das Entfernen der SWT Benutzung im Kern ermöglicht es mehr JUnit Testfälle zu schreiben. Die Entfernung des `IActivityDataObject`s musste leider vertagt werden, sie hat dennoch bei der Aufdeckung eines Problems in der Architektur beigetragen.

5 Dokumentation

Ein weiterer sowie der letzte Aspekt meiner Arbeit war die Verbesserung bzw. Erstellung von Dokumentation. Dabei handelte es sich zum einen um Dokumentation im Quelltext zur Beschreibung der Architektur und zum anderen um die Dokumentation der Prozesse. Die Hauptziele meiner Arbeit an der Dokumentation waren, den Einstieg in Saros und den Umstieg von SVN auf Git und Reviewboard auf Gerrit zu erleichtern und aktiven Entwicklern mit guter Architektur Dokumentation mehr Vertrauen bei Änderungen im Quelltext zu geben.

5.1 JTourBus

Der Großteil der Architektur-Beschreibung befand sich in ehemaligen Abschlußarbeiten, zum Teil auch auf der Webseite. Bei Veränderung der Architektur können Abschlußarbeiten nicht bearbeitet werden und auch die Webseite konnte nicht immer ohne größeren Aufwand aktualisiert werden. JTourBus bietet eine Alternative, die Dokumentation wird direkt im Quelltext geschrieben und auch dort betrachtet.

Meine Hoffnung war es die Dokumentation zugänglicher zu machen und dass bei Änderungen der Architektur die Dokumentation ebenfalls aktualisiert wird.

Bei der Einteilung meiner Zeit hatte ich jedoch ein Dilemma, die STF Tests liefen trotz der Architekturverbesserungen weiterhin nicht ohne Fehler und eine Dokumentation mit JTourBus erforderte eine Modernisierung der Eclipse-Erweiterung um auf Eclipse 3.7 installierbar zu sein. Das Dilemma war nun Folgendes: Was würde Entwicklern mehr Vertrauen geben? Eine gute Architektur Dokumentation oder eine zuverlässige Testsuite? Mit guter Dokumentation machen sie eventuell schneller eine Änderung, können sich dann aber nicht sicher sein, ob die Testfehler durch ihre Änderung kamen oder „normales Rauschen“ sind. Würde ich mich dafür entscheiden weiter die STF Tests zu stabilisieren und dafür die Dokumentation vernachlässigen ist es genau umgekehrt, eine Änderung zu machen dauert eventuell länger, aber eventuelle Testfehler enstanden dann auf Grund der Änderung. Der andere Teil der Frage ist, wie viel Zeit ich investieren müsste, um STF Tests stabil zu bekommen, aber auch wie viel Aufwand die Modernisierung von JTourBus brauchte. Ich hielt die Risiken bei der Arbeit an JTourBus für überschaubarer und entschied mich daher für das Schreiben von Touren für JTourBus unter der Bedingung, die Modernisierung innerhalb eines Tages zu schaffen.

5.1.1 Aktualisierung JTourBus

Die existierenden *Update Sites* funktionierten für Eclipse 3.7 nicht, das JTourBus Projekt wurde auch nicht mehr betreut. Ich entschied mich daher JTourBus in die Saros-Infrastruktur aufzunehmen. Dies bedeutete, Git als Versionsverwaltung zu benutzen, Gerrit für Durchsichten, Jenkins zum Testen und zur automatischen Erstellung einer *Update Site*.

1. Zuerst migrierte ich von SVN auf Git. Ich wunderte mich, wieso die Migration so schnell verlief. Der Grund war relativ einfach, es gab bisher nur einen einzelnen Commit in das öffentliche Repository.

2. Als nächstes legte ich ein Projekt für JTourBus in Gerrit an und kümmerte mich dann um die Integration zwischen Gerrit und Jenkins. Dies erforderte die Eclipse-Erweiterung mit Hilfe von **Ant** zu bauen, dazu kopierte ich eine in Saros verwendete *Buildfile* in die JTourBus Quellen und passte diese leicht an. Danach legte ich den **JTourBus-Gerrit** *Job* als Kopie des **Saros-Gerrit** *Jobs* an und konfigurierte ihn.

3. Nun konnte ich meine Änderung in das Gerrit System schieben, ich brauchte 17 Versuche um die Erweiterung auf unserer Infrastruktur zu bauen und zu testen. Eine Eclipse-Erweiterung mit Ant zu bauen ist kompliziert, zuerst braucht man eine Kopie von **ant4eclipse**, dann müssen beim Aufruf von Ant die Pfade von allen benötigten Eclipse-Bibliotheken angegeben werden. Es war daher einfacher, lokal eine Änderung zu machen und diese dann nach Gerrit zu schieben und auf das Resultat zu warten.

4. Das nächste Problem bekam ich bei der Ausführung der JUnit Tests. Ant hatte probiert den Test `junit.de.fu_berlin.inf.jtourbus.BusStopJavaDocTest` und nicht `de.fu_berlin.inf.jtourbus.BusStopJavaDocTest` auszuführen. Ich hatte erwartet, dass Ant den Namen des Paketes von Klassen aus den **.class** Dateien extrahieren würde, aber er wird aus dem Verzeichnis-Namen „erraten“. In diesem Fall war der Fehler in der Konfiguration des Jenkins *Job*. Die eigentliche Anpassung an Eclipse 3.7 bestand in der Erzeugung einer **MANIFEST.MF**, diese konnte Eclipse einfach für mich erzeugen. Ich habe keine Beschreibung gefunden, in welcher Version von OSGi diese Datei notwendig wurde.

5. Die letzten beiden Schritte waren, ein JTourBus Artefakt bei Aktualisierung des Git Repositories zu bauen, dazu konnte ich den JTourBus-Gerrit *Job* kopieren und anpassen und die Erstellung einer *Update Site*. Auch hier konnte ich einen existierenden *Job* kopieren und modifizieren, dieser Job kopierte das letzte erfolgreich gebaute Artefakt in

einen über HTTP erreichbaren Bereich und rief dann ein **Lua** Script auf, um die für die *Update Site* nötigen XML Dateien zu erzeugen. Ich entschied mich dafür, dieses Script zu kopieren und die Zeichenketten „DPP“ und „Saros“ durch „JTourBus “ zu ersetzen. Die Alternative war, das Script hinreichend zu parametrisieren. Ich entschied mich dagegen, da ich die Lua Sprache nicht beherrschte und unter keinen Umständen die Saros *Update Site* beschädigen wollte. Das Lua Script arbeitet in dem es eine Datei einliest und dann bestimmte Platzhalter durch den richtigen Inhalt ersetzt. Am Anfang hatte ich nicht alle nötigen Platzhalter in meiner Schablone, dies führte zu einigen Problemen. Ich fügte die neue *Update Site* zu Eclipse hinzu, jedoch wurde keine JTourBus Gruppe gefunden, ich konnte die Eclipse-Erweiterung einzeln auswählen, aber die Installation schlug fehl, da der Verweis auf das Artefakt falsch war. Ich fügte die nötigen Platzhalter in die Schablone ein und erzeugte die XML Dateien neu, leider hatte Eclipse die *Update Site* im *Cache* und die neue Datei wurde nicht vom Server geladen. Ich musste daher den Pfad auf dem Server ändern. Nachdem die Installation mit der geänderten URL funktionierte, änderte ich den Pfad auf den ursprünglichen Namen zurück.

5.1.2 Existierende Touren

Karl Beecher fügte die ersten Touren im August 2011 zu Saros hinzu. Diese blieben bis zum Frühjahr 2012 unverändert. Ich korrigierte lediglich einige Rechtschreibfehler und die Nummerierung der Haltestellen. Die Touren beschrieben einige Grundlagen, den Einladungsprozess und die grafische Oberfläche.

5.1.3 Neue Touren

Da ich meine ersten Erfahrungen mit dem `StopManager` machte, schrieb ich zuerst eine Tour über das Blockieren der Eingabe. Danach schrieb ich eine Tour zum Erstellen von Aktivitäten, da besonders in diesem Jahr die Anzahl der an einen bestimmten Nutzer gerichteten Aktivitäten zugenommen hat und dies nicht der Architektur entsprechend geschah. Eine weitere Tour schrieb ich über die richtige Implementierung einer neuen `IActivity`.

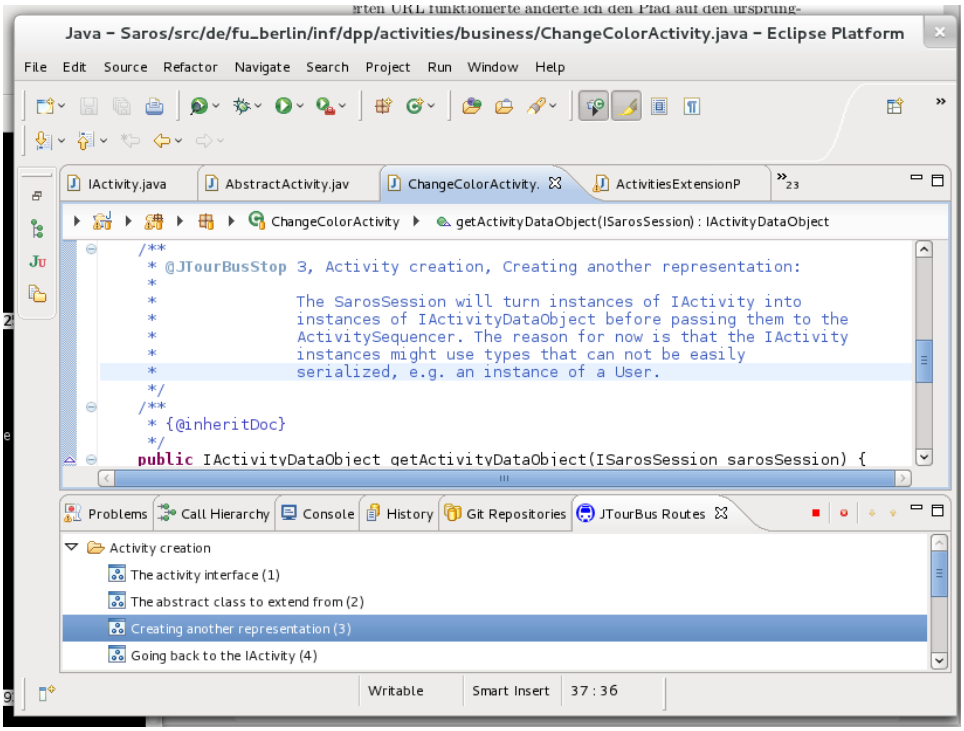


Abbildung 8: Die JTourBus Eclipse-Erweiterung

5.1.4 Fazit

Die Modernisierung von JTourBus und das Schreiben von neuen Touren geschah zum Ende meiner Arbeit, andere Entwickler haben diese Touren noch nicht benutzt und es bleibt abzuwarten, ob neue Generationen einen Nutzen aus den Touren ziehen können. Ein Beispiel für eine Haltestelle einer Tour und die Darstellung in Eclipse befindet sich in Abbildung 8.

5.2 Anleitung

Die Anleitung sollte den Umstieg von SVN auf Git und vom Reviewboard auf Gerrit erleichtern und neuen Entwicklern eine detaillierte Anleitung bieten, um alle nötigen Abhängigkeiten zu installieren, Git und Gerrit zu konfigurieren um so schnell wie möglich aktiv am Projekt teilzunehmen.

5.2.1 Wahl der Technologie

Zuerst musste ich mich für das Format und den Platz der Anleitung entscheiden. Auf Grund der beschriebenen Probleme mit dem Wiki des Fachbereiches entschied ich mich recht schnell dagegen, den Inhalt in diesem Wiki anzulegen, hinzu kam noch die Aussicht in Kürze auf ein CMS zu wechseln. Meine eigene Anforderung war es, aus einer Quelle HTML, PDF und ggf.

EPUB zu erzeugen, das PDF könnte man ausdrucken und unterwegs lesen oder beim Einrichten neben dem Computer liegen haben.

Mallard[38] und **DocBook**[39] schienen diese Anforderungen zu erfüllen und ich betrachtete beide näher. Mallard ist gedacht, um aufgabenorientierte Dokumentation, z.B. *wie installiere ich EGit*, zu schreiben und wäre daher gut für eine Anleitung mit vielen einzelnen und unabhängigen Themen geeignet. Mallard hatte jedoch keine Möglichkeit ein PDF zu erzeugen [9], daher entschied ich mich, die Anleitung mit DocBook zu schreiben.

5.2.2 Inhalt

Die Anleitung umfasste die Beschreibung der Installation von SVN, Git und JTourBus, die Erstellung eines **SSH** Schlüsselpaares für die Authentifizierung bei Git, die Erstellung eines Benutzerkontos und Konfiguration bei Gerrit, das Erstellen einer ersten Änderung mit EGit, das Eingehen auf Kommentare bei der Durchsicht und die Aktualisierung einer Änderung, das Erstellen der Release Branch und Anfänge zu Mustern und Antimustern im Saros-Quelltext. Für die einzelnen Aufgaben verwendete ich jeweils die DocBook *procedure*[40], diese ermöglichte es eine Aufgabe in aufeinander folgenden Schritten zu beschreiben. Jede Aufgabe wurde von mir in zwei inhaltlich identischen Formen aufbereitet, einmal mit Text und einmal mit Bildern. Die Variante mit Text sollte erfahrenen Entwicklern helfen so schnell wie möglich die Aufgabe durchzuführen und die andere Variante sollte besonders Neulingen helfen. Im Anhang G befindet sich ein Beispiel für eine *procedure* in den beiden Varianten.

Bei Änderungen in Eclipse, an einer Eclipse-Erweiterung oder URL, wo sich das Saros Git Repository oder ähnliches befindet, müssen sowohl der Text als auch das Bild aktualisiert werden. Ich wollte den Aufwand für die Neuerstellung der Bilder minimieren und fand die **JRuby** Integration[41]. Ich erhoffte mir die einzelnen Schritte in einem jeweils 10-zeiligen **Ruby** Script zu schreiben und automatisch Bilder zu erzeugen. Nach der Installation hatte ich jedoch Probleme von JRuby aus auf bestimmte Java Klassen zuzugreifen und konnte dieses Problem nicht in angemessener Zeit lösen. Die Alternative war es SWTBot aus Java zu benutzen, aber auch dies erschien mir nicht in angemessener Zeit machbar zu sein. Der Vorteil von Ruby lag in der interaktiven Konsole, in der ich mein Script durch Ausprobieren hätte schreiben können, in Java hingegen hätte ich ein komplettes Programm schreiben müssen.

[38] http://projectmallard.org/
[39] http://docbook.org/
[40] http://www.docbook.org/tdg/en/html/procedure.html
[41] http://swtbot.com/tools/

Nach meiner ersten Fassung hatte ich eine E-Mail an die Saros-Entwicklerliste geschickt um Kommentare zu bekommen. Die Kommentare blieben leider aus und ich sprach die Existenz meiner Anleitung bei einem der wöchentlichen Treffen an, im Anschluss waren zwei Mitglieder des Saros-Teams so freundlich und haben die Anleitung gelesen und ausprobiert. Neben einigen Problemen mit der Rechtschreibung enstand ein weiteres Problem durch eine neue Version der EGit Eclipse-Erweiterung. Der Ablauf beim *Clonen* hatte sich verändert, sodass Text und Bild nicht mehr stimmten. Das Problem löste ich, in dem ich den Text und die Bilder aktualisierte und die Versionsnummer der EGit Eclipse-Erweiterung im Text vermerkte. Die von mir gewünschte automatische Erstellung der Bilder hätte in diesem Fall nicht geholfen und hätte aktualisiert werden müssen. Das Aktualisieren der Bilder ging jedoch recht schnell, sehr wahrscheinlich schneller als die Wiedereinarbeitung in JRuby.

5.2.3 Jenkins und Gerrit Integration

Die XML Dateien der Anleitung befinden sich in einem Git Repository[42], Änderungen gehen über das Gerrit System des Projektes. Zusätzlich wird jede Nacht eine aktuelle Version gebaut und die HTML und PDF Version in einen per HTTP erreichbaren Bereich[43] kopiert. Die Saros-Webseite hat an den nötigen Punkten Verweise auf die Dateien.

Zum Erstellen von HTML und PDF habe ich eine **Makefile** geschrieben, zur Erzeugung des PDFs habe ich mich für **dblatex**[44] entschieden und für die HTML Variante nehme ich **xsltproc** und die *Stylesheets* von DocBook.

5.2.4 Fazit

Während meiner Arbeit stieg jedoch nur eine weitere technische Person in das Projekt ein. Diese war in der Lage, sich mit Hilfe der Anleitung bei Gerrit zu registrieren und eine Durchsicht zu starten. Die Anleitung war ausführlich und für das Saros-Team verständlich.

[42] http://dpp.git.sourceforge.net/git/gitweb.cgi?p=dpp/saros-developer-guide;a=summary

[43] http://saros-build.imp.fu-berlin.de/sarosGettingStarted/

[44] http://dblatex.sourceforge.net/

6 Rückblick und Ausblick

> Alles nimmt ein gutes Ende für den, der warten kann.
>
> Leo N. Tolstoi

6.1 Rückblick

Viele der in Kapitel 2 beschriebenen Probleme wurden während meiner Arbeit von mir oder anderen bearbeitet und ich gebe hier einen kurzen Überblick.

- Es gab Anfänge von einer Spezifikation auf der Saros-Webseite, einzelne Entscheidungen können dort festgehalten werden.
- Stefan Rossbach und ich ersetzten an vielen Stellen die Verwendung von spezifischen Implementierungen in den Signaturen durch die Verwendung von Schnittstellen.
- Die Verwendung hierachischer PicoContainer war eine Alternative zur vorgeschlagenen Modularisierung, zumindest die Trennung der Sichtbarkeit kann auf diese Weise realisiert werden.
- An einigen Stellen habe ich durch richtige Verwendung des PicoContainers handgeschriebenen Quelltext zum Auflösen von Abhängigkeiten entfernen können.
- Ich habe alle Sicherheitsupdates auf der Infrastruktur installiert, die `SarosTech` Seite im Wiki aktualisiert und durch eine Sektion mit durchgeführten Änderungen erweitert.
- Der Einstieg in das Projekt sollte durch die detaillierte Anleitung einfacher sein.
- Es war beabsichtigt die Website zu restrukturieren und auf eine andere Technologie umzustellen.
- Ich habe angefangen nicht mehr benutzten Quelltext zu entfernen, dabei wurden einige Fehler aufgedeckt und korrigiert.
- Einige der Brüche mit der Architektur wurden durch mich korrigiert, in diesen Fällen ist es jetzt einfacher die Klassen richtig zu benutzen.
- Die Umstellung auf Gerrit und Git hatte den Durchsichtsprozess erheblich verbessert, besonders die Integration zwischen Gerrit und Jenkins vermeidet viele Fehler.

Ich hatte die Hoffnung durch meine Architekturkorrekturen die STF Tests stabil zu bekommen, diese Hoffnung hat sich leider nicht bestätigt. Auf der anderen Seite war die Umstellung auf Git und Gerrit und die Integration zwischen Gerrit und Jenkins eine sehr positive Prozessänderung, viele der Probleme mit SVN und Reviewboard sind verschwunden, das Arbeiten an mehr als einer Änderung ist nun einfach, das Testen einer anderen Änderung ging schneller und einfacher. Das Erstellen der Anleitung und das Beschreiben der Architektur mit JTourBus sollten neuen und existierenden Entwicklern helfen effektiver zu arbeiten. Ich glaube daher, das Ziel meiner Arbeit erreicht zu haben.

6.2 Ausblick

Das Ziel des Projektes sollte sein, Mitentwickler außerhalb des Fachbereiches zu finden und länger an das Projekt zu binden. So können Rollen zur Wartung der Infrastruktur, zum Durchsehen der Änderungen besser besetzt werden. Auch auf technischer Seite existieren viele Möglichkeiten zur Verbesserung von Saros und ich beschreibe diese im Anhang H näher. Ich wünsche dem Saros-Projekt noch viel Erfolg.

A Antwort zum Öffnen eines Ports

Dies ist die Antwort der Technik bezüglich der Freigabe eines Ports.

```
Ja, Ablaufdatum ist konsistent zu den anderen Freigaben für saros-build gesetzt:

<link src="DEFAULT" dst="saros_build">
  <description> Florian Thiel, RT #107377 und #108535</description>
        <use clearance="jabber">
                        <valid from="2010-03-25T14:47:00" to="2014-01-31T23:59:00" auto="true"/>
        </use>
        <use clearance="socks5">
                        <valid from="2010-03-25T14:47:00" to="2014-01-31T23:59:00" auto="true"/>
        </use>
        <use clearance="stun">
                        <valid from="2010-03-25T14:47:00" to="2014-01-31T23:59:00" auto="true"/>
        </use>
        <use clearance="web">
                        <valid from="2010-05-18T21:00:00" to="2014-01-31T23:59:00" auto="true"/>
        </use>
        <use clearance="gerrit">
                        <valid from="2012-04-10T10:00:00" to="2014-01-31T23:59:00" auto="true"/>
        </use>
</link>
```

B Anatomie eines Git Commits

Dies beschreibt die Anatomie eines Git Commits auf der untersten Ebene und soll dem Verständnis von Git dienen. Ich beginne mit einer Commit Meldung:

```
commit 824ff128d2a6c7b7586b02cb7717af580ff3f6b5
Author: Holger Hans Peter Freyther <holger@freyther.de>
Date:   Sun Jun 17 19:32:20 2012 +0200

    text...
```

Es handelt sich um das Git Objekt **824ff128d2a6c7b7586b02cb7717af580ff3f6b5** und es ist vom Typ `commit`. Mit dem `$ git cat-file commit 824ff128` Kommando kann man den Commit betrachten. Unter anderem erscheint dort eine Nummer für ein Objekt vom Typ `tree`.

```
$ git cat-file commit
tree f4c759093a7cccd2435c48f0c2aa3a543d05d92f
parent 0c27d8552efdef13745c13077712708d819b1fec
author Holger Hans Peter Freyther <holger@freyther.de> 1339954340 +0200
committer Holger Hans Peter Freyther <holger@freyther.de> 1339954340 +0200

text....
```

Das `tree` Objekt beschreibt einen Zustand des Wurzelverzeichnises, in diesem Fall sah dieser so aus:

```
$ git cat-file -p f4c759093a7cccd2435c48f0c2aa3a543d05d92f
100644 blob 93b6499b7b18b9fc9826eb32a181ddaab2e920d0    .gitignore
040000 tree cd6ea66501dce6b5d673e5955172948fd647b505    de.fu_berlin.inf.dpp.feature
040000 tree 09afab6292ca1cc462899bf72532103f0e758ebf    de.fu_berlin.inf.dpp.ui.widgetGallery
040000 tree 18949f4228c08930b131252877d1847ec4348b16    de.fu_berlin.inf.dpp.update
040000 tree 460026ede8ae8deff85cdc3d50afdd5200e8f3b7    de.fu_berlin.inf.dpp.whiteboard
040000 tree 20cefcd90dcd074912f07cb9b2b7fd8bd5b85aa3    de.fu_berlin.inf.dpp
040000 tree 55b2aeb8acf6bf4e31d19d34575d61be2a9f290b    de.fu_berlin.inf.nebula
040000 tree d82a2a3359a65079b24e50d0893b275fce0941e0    misc
```

In dem `tree` Objekt des Wurzelverzeichnisses existieren Verweise auf weitere `tree` Objekte und ein neues Objekt vom Typ `blob`, dieser `blob` beinhaltet den Inhalt der Datei. Für jeden Eintrag in einem `tree` werden die Unix Zugriffsrechte gespeichert.

Bei einer Änderung von *.gitignore* würde sich die Prüfsumme der Datei ändern, damit ändert sich der Inhalt des `tree` Objektes des Wurzelverzeichnisses und hat daher ebenfalls eine andere Prüfsumme. Ein `commit` Objekt würde sich dann auf das neue `tree` Objekt beziehen.

C SVN Benutzername auf Name und E-Mail

Die Abbildung von SVN Benutzername auf Name und E-Mail Adresse aus öffentlich zugänglichen Daten.

```
coezbek = Christopher Oezbek <coezbek@users.sourceforge.net>
kargor = Stefan Rossbach <kargor@users.sourceforge.net>
nudelfish = nudelfish <nudelfish@users.sourceforge.net>
chris_fu = Christoph Jacob <chris_fu@users.sourceforge.net>
ahaferburg = A. Haferburg <ahaferburg@users.sourceforge.net>
k_beecher = Karl Beecher <k_beecher@users.sourceforge.net>
bkahlert = Björn Kahlert <bkahlert@users.sourceforge.net>
marrin = Marc 'BlackJack' Rintsch <marrin@users.sourceforge.net>
orieger = Oliver Rieger <orieger@users.sourceforge.net>
s-ziller = Sebastian Ziller <s-ziller@users.sourceforge.net>
freyther = Holger Freyther <freyther@users.sourceforge.net>
netcorps = Alexander Waldmann <netcorps@users.sourceforge.net>
Arbosh = Arbosh <Arbosh@users.sourceforge.net>
ornis = Bjoern <ornis@users.sourceforge.net>
djemili = Riad Djemili <djemili@users.sourceforge.net>
florianthiel = Florian Thiel <florianthiel@users.sourceforge.net>
donut87 = Donut <donut87@users.sourceforge.net>
szuecs = Sandor Szuecs <szuecs@users.sourceforge.net>
ldohrmann = Lisa Dohrmann <ldohrmann@users.sourceforge.net>
karlheld = Karl Held <karlheld@users.sourceforge.net>
sotitas = Tas Sóti <sotitas@users.sourceforge.net>
translatemyname = Hendrik <translatemyname@users.sourceforge.net>
```

```
s-lau = Stephan Lau <s-lau@users.sourceforge.net>
p-cordes = Philipp Cordes <p-cordes@users.sourceforge.net>
sbauch = sbauch <sbauch@users.sourceforge.net>
meike11 = meike <meike11@users.sourceforge.net>
mvhoffen = mvh <mvhoffen@users.sourceforge.net>
patbit = Patrick Bitterling <patbit@users.sourceforge.net>
arbosh = Michael Jurke <arbosh@users.sourceforge.net>
hstaib = hstaib <hstaib@users.sourceforge.net>
alxrsaros = alxrsaros <alxrsaros@users.sourceforge.net>
nwarnatsch = Norman Warnatsch <nwarnatsch@users.sourceforge.net>
starkmann = Eike Starkman <starkmann@users.sourceforge.net>
testvogel = ologa <testvogel@users.sourceforge.net>
uerdogan = Umut <uerdogan@users.sourceforge.net>
franzzieris = Franz Zieris <franzzieris@users.sourceforge.net>
belousow = Slawa Belousow <belousow@users.sourceforge.net>
marcus-fu = marcus-fu <marcus-fu@users.sourceforge.net>
andreas-fu = andreas-fu <andreas-fu@users.sourceforge.net>
fpuetz0 = fpuetz <fpuetz0@users.sourceforge.net>
kbeecher = Karl Beecher <kbeecher@users.sourceforge.net>
wojtus = wojtek <wojtus@users.sourceforge.net>
```

D svn2git Regeln

Die bei der Git Umstellung benutzten Regeln zum Aufspalten in verschiedene Repositories und zum Zusammenführen der verschiedenen *Release Branches*.

```
#
# Declare the repositories we know about:
#

create repository saros-plugin
end repository
create repository saros-statisticserver
end repository
create repository saros-updatesitetools
end repository
create repository saros-stf
end repository

#
# Declare the rules
# Note: rules must end in a slash
#

# saros-statisticserver
match /trunk/de.fu-berlin.inf.SarosStatisticServer/
  repository saros-statisticserver
  branch master
end match

# saros-updatesitetools
```

```
match /trunk/UpdateSiteTools/
  repository saros-updatesitetools
  branch master
end match

# updatesitetools
match /trunk/(update_sf.sh|updateSiteXML.py)
  repository saros-updatesitetools
  prefix \1
  branch master
end match

match /branches/UpdateSiteToolsForStableReleases/
  repository saros-updatesitetools
  branch stableReleases
end match


# saros-stf
match /stf_tests/
  repository saros-stf
  branch master
end match

# saros-plugin
match /trunk/dpp/
  repository saros-plugin
  branch master
end match

match /branches/concurrents/Saros/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp/
  branch development/concurrents
end match

match /branches/orieger/smack3.5.0/Saros/
  repository saros-plugin
  branch development/smack3.5.0
end match

# r245 added .classpath and .properties => ignore
match /branches/orieger/smack3.5.0/
end match

# r3009 added stun-config.xml => ignore
match /branches/ecfIntegration/
end match

# wrong branch name by orieger
match /branches/https:/dpp.svn.sourceforge.net/svnroot/dpp/branches
    /10.11.26.r2744/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp/
```

```
  branch release/10.11.26.r2744
end match

# wrong branch name by sbauch (r2093)
match /branches/https:/svn.sourceforge.net/svnroot/dpp/branches/10.3.26.
    r2091/de.fu_berlin.inf.dpp/
  prefix de.fu_berlin.inf.dpp/
  repository saros-plugin
  branch release/10.3.26.r2091
end match

# Only copied dpp/de.fu_berlin.inf.dpp/
match /branches/(9.5.7|9.5.28|9.6.18|9.7.10|9.7.31.r1559|9.8.21.r1660
    |9.9.11.r1706|9.10.2.r1803|9.10.30.r1833|9.12.04|10.1.29.r1966
    |10.2.26.r2010|10.3.26.r2091|10.4.14.r2128|10.5.28.r2173|10.6.11.
    r2223|10.6.25.r2236|10.7.30.r2303|10.8.27.r2325|10.10.01.r2531
    |10.10.29.r2640|11.1.7.r2897|11.1.28.r2959|10.11.26.r2744|10.3.26.
    r2091|9.10.2.r1903)/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp/
  branch release/\1
end match

match /branches/(chatView|teles|consistencyWatchdog|macrian|macrian2|
    net_streaminterface|net_voip|screensharing|net_streaminterface_trunk|
    whiteboard|net_refactoring|net_refactor|vcsSync|svnIntegration|
    tmp_haferburg|tmp_haferburg2|distrib_debug|ecf_integration|
    multipleProjects|viewMerging|whiteboard_batik|de.fu_berlin.inf.dpp|
    usability.waldmann|net_voip_xmpp|net_voip2|dpp_light|voip_ana_ext)/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp/
  branch development/\1
end match

match /branches/streaming_project_experimental/
  repository saros-plugin
  branch development/streaming_project_experimental
end match

match /branches/de.fu_berlin.inf.dpp.whiteboard/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp.whiteboard/
  branch development/whiteboard-trunk-preparation
end match

match /de.fu_berlin.inf.nebula/
  repository saros-plugin
  prefix de.fu_berlin.inf.nebula
  branch development/nebula-preparation
end match


##############################
#
```

```
# Releases to merge things...
#
############################

# Need to merge different parts of a release together
match /branches/dpp.whiteboard_11.1.7.r2965/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp.whiteboard/
  branch release/11.1.7.r2965
end match

# Merging separate releases into a single branch
# 11.2.25 release
match /branches/dpp_11.2.25.r3105/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp/
  branch release/11.2.25.r3105
end match
match /branches/dpp.whiteboard_11.2.25.r3105/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp.whiteboard/
  branch release/11.2.25.r3105
end match

# 11.3.25 release
match /branches/dpp_11.3.25.r3201/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp/
  branch release/11.3.25.r3201
end match
match /branches/dpp.whiteboard_11.3.25.r3201/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp.whiteboard/
  branch release/11.3.25.r3201
end match

# dpp_11.5.6.r3294
match /branches/dpp_11.5.6.r3294/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp/
  branch release/11.5.6.r3294
end match
match /trunk/dpp.whiteboard_11.5.6.r3294/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp.whiteboard/
  branch release/11.5.6.r3294
end match
match /branches/dpp.whiteboard_11.5.6.r3294/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp.whiteboard/
  branch release/11.5.6.r3294
end match

# dpp_11.7.1.r3426
```

```
match /branches/dpp_11.7.1.r3426/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp/
  branch release/11.7.1.r3426
end match
match /branches/dpp.whiteboard_11.7.1.r3426/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp.whiteboard/
  branch release/11.7.1.r3426
end match

# dpp_11.7.29.r3479
match /branches/dpp_11.7.29.r3479/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp/
  branch release/11.7.29.r3479
end match
match /branches/dpp.whiteboard_11.7.29.r3479/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp.whiteboard/
  branch release/11.7.29.r3479
end match

# 11.9.30.r3567
match /branches/(dpp_10.9.30.r3567|dpp_11.9.30.r3567)/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp/
  branch release/11.9.30.r3567
end match
match /branches/dpp.whiteboard_11.9.30.r3567/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp.whiteboard/
  branch release/11.9.30.r3567
end match

# 11.12.9.r3685
match /branches/dpp_11.12.9.r3685/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp/
  branch release/11.12.9.r3685
end match
match /branches/dpp.whiteboard_11.12.9.r3685/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp.whiteboard/
  branch release/11.12.9.r3685
end match

# 12.2.24.r3716
match /branches/dpp_12.2.24.r3716/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp/
  branch release/12.2.24.r3716
end match
match /branches/dpp.whiteboard_12.2.24.r3716/
```

```
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp.whiteboard/
  branch release/12.2.24.r3716
end match
match /branches/nebula_12.2.24.r3716/
  repository saros-plugin
  prefix de.fu_berlin.inf.nebula/
  branch release/12.2.24.r3716
end match

# 12.3.30.r3893
match /branches/dpp_12.3.30.r3893/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp/
  branch release/12.3.30.r3893
end match
match /branches/dpp.whiteboard_12.3.30.r3893/
  repository saros-plugin
  prefix de.fu_berlin.inf.dpp.whiteboard/
  branch release/12.3.30.r3893
end match
match /branches/nebula_12.3.30.r3893/
  repository saros-plugin
  prefix de.fu_berlin.inf.nebula/
  branch release/12.3.30.r3893
end match
#match /
#end match

# Important:
# Subversion doesn't understand the Git concept of tags
# In Subversion, tags are really branches
#
# Only a post-processing (i.e., after converting to Git) of the tag
# branches can we be sure that a tag wasn't moved or changed from the
# branch it was copied from
#
# This rule will create tags that don't exist in any of the
# branches. It's not what you want.
# See the merged-branches-tags.rules file
match /tags/([^/]+)/
  repository saros-plugin
  branch refs/tags/\1
end match
```

E svn2git Änderung

Die Änderung um beim Erstellen einer Branch keine Dateien zu löschen.

```
From 7153aeb9897827e46ec49233c6831280f10fd1dd Mon Sep 17 00:00:00 2001
From: Holger Hans Peter Freyther <zecke@selfish.org>
Date: Mon, 30 Apr 2012 17:05:10 +0200
```

```
Subject: [PATCH] Fix the 'branching' e.g. on dpp_light but other branches
    as
 well

It was possible that due the code I added earlier that a parent
might have been dropped from the history. On the creation on some
branches it is possible that different revisions are copied over
the place and that the parent commit was lost. On the other hand
we want the branches to be 'complete' (include nebula, whiteboard,
etc. from the point where we branches) fix it by looking at when
the branch was created and then use the 'from' for the first parent.
---
 src/repository.cpp |    9 +++++-----
 1 file changed, 4 insertions(+), 5 deletions(-)

diff --git a/src/repository.cpp b/src/repository.cpp
index bc790a1..7121c5f 100644
--- a/src/repository.cpp
+++ b/src/repository.cpp
@@ -742,10 +742,6 @@ void Repository::Transaction::commit()
         merges.pop_back();
         qWarning() << "WARN: Discarding all but the highest merge point
             as a workaround for cvs2svn created branch/tag"
                        << "Discarded marks:" << merges;
-    } else if (merges.count() == 1 && (branch.contains("release") ||
    revnum == 2095 || branch.contains("development"))) {
-        QByteArray m = " :" + QByteArray::number(merges.first());
-        desc += m;
-        repository->fastImport.write("from" + m + "\n");
     } else {
         foreach (const int merge, merges) {
             if (merge == parentmark) {
@@ -766,7 +762,10 @@ void Repository::Transaction::commit()

             QByteArray m = " :" + QByteArray::number(merge);
             desc += m;
-            repository->fastImport.write("merge" + m + "\n");
+            if (i == 1 && br.created == revnum)
+                repository->fastImport.write("from" + m + "\n");
+            else
+                repository->fastImport.write("merge" + m + "\n");
         }
     }
     // write the file deletions
--
1.7.10.4
```

F Entfernen von Methoden und Klassen

Eine Liste von Git Commit Meldungen zum Entfernen von verwaisten Methoden und Klassen aus Saros.

```
commit ce10b0c254baf7c332eac0bca746cca1195074a9
Author: Holger Freyther <freyther@users.sourceforge.net>
Date:   Wed Mar 21 13:11:51 2012 +0000

    [INTERNAL] Remove the unused and leaking activity history feature

    The history list in ActivitySequencer.ActivityQueue was growing
    over the lifetime of a session without an upper bound. The list
    was kept for a feature to request a client to resend some activities
    in case of a sequence error. The last call was removed recently
    in r3806 but the last caller was not called.

    Change-Id: Iba0a22f138a1628082192345e2b3dd9e9706bb42

Notes:
    svn path=/trunk/dpp/; revision=3865
```

```
commit 0dfb8ee458820362119f6021c74bf75a662df6b9
Author: Holger Freyther <freyther@users.sourceforge.net>
Date:   Sat Mar 10 22:28:50 2012 +0000

    [INTERNAL] Remove the unused UncloseableInputStream class

    Noticed while playing with Moose (moosetechnology.org).

    Change-Id: Ibfead06e01cd48aadece5fa3423bf638b71b5896

Notes:
    svn path=/trunk/dpp/; revision=3819
```

```
commit 3e65dcd57005a8f4c0875d4eb382f333295fb65b
Author: Holger Freyther <freyther@users.sourceforge.net>
Date:   Sat Mar 10 22:28:22 2012 +0000

    [INTERNAL] Remove the StreamServiceManager Invitation and Negotiation

    The subclasses of InvitationProcess and ProjectNegotiation do not use
    the StreamServiceManager. Remove unused functionality.

    Change-Id: Id4c92415bf5c261beba9e6cbe119bcfc8d432b7f

Notes:
    svn path=/trunk/dpp/; revision=3818
```

```
commit 58fa842b5a394f3efd2b5512dfb35a50565c783f
Author: Holger Freyther <freyther@users.sourceforge.net>
Date:   Sat Mar 10 16:37:38 2012 +0000

    [INTERNAL] Remove unused loggers

    Change-Id: Ia57a487223c05b721245991d3cac90c4ab536aca

Notes:
    svn path=/trunk/dpp/; revision=3815
```

```
commit 87dfe3fea8a5609f17169d862f7e0666c15bb611
Author: Holger Freyther <freyther@users.sourceforge.net>
Date:   Sat Mar 3 18:25:46 2012 +0000

    [INTERNAL] Remove NetworkSimulator.getLastError and callers

    NetworkSimulator stopped using threads in r2048 and at the same time
    stopped recording exceptions by this thread. Findbugs complained that
    the error field in NetworkSimulator is not written to. Remove the field,
    the method and the callers. All Exceptions will be thrown directly and
    JUnit will properly handle them.

    Change-Id: Iec6bbb149c57d396d0c3f3038b1e602cc90ff3e5

Notes:
    svn path=/trunk/dpp/; revision=3778
```

```
commit 7037dee1fffd0352ca5cc9699275591295d13973
Author: Holger Freyther <freyther@users.sourceforge.net>
Date:   Sat Mar 3 18:25:09 2012 +0000

    [NOP] Remove IFileTransferCallback/AbstractFileTransferCallback

    There are no instantiable subclasses in the source tree, the last
    usage was removed in r2913. The class has been with the project from
    r4 on.

    Change-Id: I4265c2ba6c68811dca986deb5b8741658a535f7a

Notes:
    svn path=/trunk/dpp/; revision=3777
```

```
commit db1eda4c5de17b321ddd0ba92de023b5f4bbc474
Author: Holger Freyther <freyther@users.sourceforge.net>
Date:   Sat Mar 3 18:24:39 2012 +0000

    [NOP] Remove IDataReceiver

    This class was added in r667 (2009) and was made obsolete in
    r1798 by moving to XMPPReceiver for handling packets and connections.

    Change-Id: Ifa1bdc7e0dda7235fde57b35d6fcfda3d87aa60f

Notes:
    svn path=/trunk/dpp/; revision=3776
```

```
commit 2cf86c6e86643de72cef5d3cfee0d6db24861391
Author: Holger Freyther <freyther@users.sourceforge.net>
Date:   Sat Mar 3 18:24:05 2012 +0000
```

```
    [NOP] Remove Utils.getFreePort that luckily is not used anymore.

    The last consumer was the JingleTransferSession. This method is
    dangerous as there is no gurantee that the port will still be
    available when the caller will try to create a socket.

    Change-Id: I004de69303ce28136757e4e2efcfcfa899b7668c

Notes:
    svn path=/trunk/dpp/; revision=3775
```

```
commit 68faa21cbdb24b41b10741efc254423ecf0f1c5f
Author: Holger Freyther <freyther@users.sourceforge.net>
Date:   Sat Mar 3 18:23:36 2012 +0000

    [NOP] Supplier was added but never used

    Suppler got added in r1349 (2009) but was never used by anything,
    the original problem was addressed differently by Sandor in
    r1811 of the 9.10.2.1803 branch. He made the cache to access the
    DiscoveryInfo in the DiscoveryManager synchronized and accessed
    it by JID.

    Change-Id: I9867ca3656d399b1f09816ed2ed2c883b19b538b

Notes:
    svn path=/trunk/dpp/; revision=3774
```

```
commit 9af6b2837a9f5978fb0dc1c7e54600841d202808
Author: Holger Freyther <freyther@users.sourceforge.net>
Date:   Fri Mar 2 10:58:39 2012 +0000

    [NOP] Remove unused iterator classes

    Both classes were added in r1282 (2009), gained unit tests
    in r2125 (2010) but were never used by client code. It is
    not likely that they will be used soon.

    Change-Id: Ib98dc7a109a8ac89a6379e75732b17695c61a42d

Notes:
    svn path=/trunk/dpp/; revision=3764
```

```
commit 4d7f71fda7ac48811a9ab64b4edd02e1718bd45b
Author: Holger Freyther <freyther@users.sourceforge.net>
Date:   Fri Mar 2 10:57:56 2012 +0000

    [NOP] Remove unused class UserCancellationException.

    The class got introduced in r1798 and should have been removed
    with r1832 and the move to SarosCancellationException.
```

```
    Change-Id: Ic622fa3d360d42a29948a5cc3432251a11e29084

Notes:
    svn path=/trunk/dpp/; revision=3763
```

```
commit 56702d521cf39f274eec810fb6dfc00619756c87
Author: Holger Freyther <freyther@users.sourceforge.net>
Date:   Fri Mar 2 10:57:26 2012 +0000

    [NOP] Remove unused RequestForFileListExtension class

    The last usage was removed in r1798.

    Change-Id: I6ae8658b9ad5e6506c1bd2dff50fcf6c3b4e948a

Notes:
    svn path=/trunk/dpp/; revision=3762
```

```
commit 962d25f53cb60a5de454241f6f2bc9e385b6850d
Author: Holger Freyther <freyther@users.sourceforge.net>
Date:   Fri Mar 2 10:56:53 2012 +0000

    [NOP] Remove unused DataConnection class

    This class was introduced in r2223. This commit is a merge of the
    net_refactoring branch. The class was never used.

    Change-Id: I7e4b30c69d343f25f470bc0cd390de8d995bafa1

Notes:
    svn path=/trunk/dpp/; revision=3761
```

```
commit f82fd893d086913738f71da3140de48f7bb3739f
Author: Holger Freyther <freyther@users.sourceforge.net>
Date:   Fri Mar 2 10:56:21 2012 +0000

    [NOP] Remove unused XMMPPAccountUtils class

    Commit r3700 removed the testcases and usages of this class, let it
    rest in peace now.

    Change-Id: I183c29db6fdd486b455e37d4eefef739d06b2b87

Notes:
    svn path=/trunk/dpp/; revision=3760
```

```
commit eaabbc253f5e29a5a9647b21926f34f4248fe96e
Author: Holger Freyther <freyther@users.sourceforge.net>
Date:   Fri Mar 2 10:55:52 2012 +0000

    [NOP] Remove EditorUtils as it is not used
```

```
    The last usage was removed in r1742.

    Change-Id: I3df4bffd5712a3ac7156eca0854f61cdcbaf544d

Notes:
    svn path=/trunk/dpp/; revision=3759
```

```
commit d5f9ebbef87b7271b6d1fe08bf04d73ae2573e42
Author: Holger Freyther <freyther@users.sourceforge.net>
Date:   Fri Mar 2 10:55:23 2012 +0000

    [NOP] Remove the TimestampFactory interface and implementation

    TimestampFactory/JuputerTimestampFactory got added in r160, the code
    that would instantiate that factory to create a timestamp was added
    in r172 but it has always been commented out.

    Change-Id: I524cae411d6dc68da93bf762e74ccc10ba7854cb

Notes:
    svn path=/trunk/dpp/; revision=3758
```

```
commit fc77f3269a870ae402e1cb21dd339006470fa0c2
Author: Holger Freyther <freyther@users.sourceforge.net>
Date:   Tue Feb 21 20:32:31 2012 +0000

    [NOP] XMPPTransmitter.inject is dead code, remove it

    In general we would inject into this class using the PicoContainer.

    Change-Id: I170b574998c3aad01f0e3938f818bc595f1222d6

Notes:
    svn path=/trunk/dpp/; revision=3735
```

```
commit 3b21eea60323d49463c72a8add8c4b4f09cf2ae5
Author: Holger Freyther <freyther@users.sourceforge.net>
Date:   Tue Feb 21 20:31:54 2012 +0000

    [NOP] Remove EclipseHelper from the code

    The EclipseHelper class was introduced in r3356 as part of the
    JUNIT work. This work has been removed from the tree in r3699. Remove
    the EclipseHelper as the indirection was only needed for the JUNIT
    test framework.

    Change-Id: I26ebc8712100bcf3b0bc0683adb4703bffb0b40d

Notes:
    svn path=/trunk/dpp/; revision=3734
```

```
commit 3374233cc0ce963c953e8efbb80f08e5cd5fda86
Author: Holger Freyther <freyther@users.sourceforge.net>
Date:   Thu Mar 8 17:12:31 2012 +0000

    [CHANGE] Remove broken and unused revert handling code

    The revert handling code got disabled in SVN revision r2969 when adding
    the possibility to add projects to an existing session. The attempt to
    re-enable this change leads to modification circles. In case Bob is
    following Alice and Alice is closing an editor, the code will trigger
    on both Alice and Bob, Alice and Bob will now respond to the
    modification each other made and will continue to do so.

    The right behavior for following should be discussed because right now
    both Alice and Bob will be asked if they want to save the modification.

    Change-Id: If763ffe419838bc0ff5574114bfcc642f52b7bc4

Notes:
    svn path=/trunk/dpp/; revision=3796
```

G DocBook Prozedur Beispiel

Ein Beispiel für die Verwendung der DocBook *procedure* in der Saros-Anleitung. Zwei Prozeduren mit einzelnen Schritten.

```
<procedure><title>Git clone Saros</title>
<step>
  <title>Begin</title>
  <para><menuchoice>
    <guimenu>File</guimenu><guimenuitem>Import</guimenuitem>
  </menuchoice></para>
</step>
<step>
  <title>Select import from Git</title>
  <para>Select <menuchoice>
<guimenu>Git</guimenu><guimenuitem>Projects from Git</guimenuitem>
  </menuchoice> and press <guibutton>Next</guibutton>.</para>
</step>
<step>
  <title>Select Repository Source</title>
  <para>Select <guimenuitem>URI</guimenuitem> and press <guibutton>Next
      </guibutton>.
  </para>
</step>
<step>
  <title>Source Git repository</title>
  <para>Fill in <emphasis>dpp.git.sourceforge.net</emphasis> as Host,
      use
  <emphasis>gitroot/dpp/saros</emphasis> as repository path and select
  <emphasis>git</emphasis> as the protocol and finish by pressing the
```

```
  <guibutton>Next</guibutton> button.</para>
</step>
<step>
  <title>Select the branch</title>
  <para>Select the <emphasis>master</emphasis> branch and continue by
  pressing <guibutton>Next</guibutton>.</para>
</step>
<step>
  <title>Select local destination</title>
  <para>Decide where to store the cloned Git repository. The proposed
  directory should work and press <guibutton>Next</guibutton>.</para>
</step>
<step>
  <title>Cloning the repository</title>
  <para>Progress of downloading the repository is shown.</para>
</step>
<step>
  <title>Select project directory</title>
  <para>Select <emphasis>Import existing projects</emphasis> and import
       the
  top level directory <emphasis>Working Directory</emphasis> and
      continue
  using the <guibutton>Next</guibutton>.</para>
</step>
<step>
  <title>Import projects</title>
  <para>Select at least the <emphasis>de.fu_berlin.inf.nebula</emphasis
      > and
  <emphasis>Saros</emphasis> to import.</para>
</step>
<step>
  <title>Switch to <emphasis>Git Perspective</emphasis></title>
  <para>Clicking on
  <menuchoice><guimenu>Window</guimenu><guimenu>Open Perspective</
      guimenu>
  <guimenuitem>Other</guimenuitem></menuchoice> and select the <
      emphasis>
  Git Perspective</emphasis>.</para>
</step>
<step>
  <title>Enter Gerrit Configuration</title>
  <para>Unfold the <emphasis>Saros</emphasis>, <emphasis>Remotes</
      emphasis>
  and <emphasis>origin</emphasis> folder. Use the context menu and
      select
  <emphasis>Gerrit Configuration</emphasis>.
  </para>
</step>
<step>
  <title>Configure Gerrit</title>
  <para>Use <emphasis>ssh://saros-build.imp.fu-berlin.de:29418/saros.
      git</emphasis> as
  the <emphasis>Push URI</emphasis> and <emphasis>refs/for/master</
      emphasis>
```

```
  as the <emphasis>Destination branch</emphasis>.
  </para>
</step>
<step>
  <title>Verify EGit is enabled</title>
  <para>Switch to the <emphasis>Java Perspective</emphasis> and use the
      context
  menu in the <emphasis>Package Explorer</emphasis> and verify that the
      <guimenu>Team
  </guimenu> contains options for commit.</para>
</step>
</procedure>
<procedure><title>Git clone Saros</title>
<step>
  <title>Begin</title>
  <para><mediaobject><imageobject>
    <imagedata fileref="images/egit-clone/01_file_import.png" />
  </imageobject></mediaobject></para>
</step>
<step>
  <title>Select import from Git</title>
  <para><mediaobject><imageobject>
    <imagedata fileref="images/egit-clone/02_file_import_git.png" />
  </imageobject></mediaobject></para>
</step>
<step>
  <title>Select Repository Source</title>
  <para><mediaobject><imageobject>
    <imagedata fileref="images/egit-clone/03_file_import_uri.png" />
  </imageobject></mediaobject></para>
</step>
<step>
  <title>Source Git Repository</title>
  <para><mediaobject><imageobject>
    <imagedata fileref="images/egit-clone/04_file_import_git.png" />
  </imageobject></mediaobject></para>
  <para>Host: <emphasis>dpp.git.sourceforge.net</emphasis>,
  Path: <emphasis>gitroot/dpp/saros</emphasis></para>
</step>
<step>
  <title>Select the branch</title>
  <para><mediaobject><imageobject>
    <imagedata fileref="images/egit-clone/05_git_select_master.png" />
  </imageobject></mediaobject></para>
</step>
<step>
  <title>Select local destination</title>
  <para><mediaobject><imageobject>
    <imagedata fileref="images/egit-clone/06_git_select_location.png" /
       >
  </imageobject></mediaobject></para>
</step>
<step>
  <title>Cloning the repository</title>
```

```
  <para><mediaobject><imageobject>
    <imagedata fileref="images/egit-clone/07_git_clone.png" />
  </imageobject></mediaobject></para>
</step>
<step>
  <title>Select project directory</title>
  <para><mediaobject><imageobject>
    <imagedata fileref="images/egit-clone/08_import_projects.png" />
  </imageobject></mediaobject></para>
</step>
<step>
  <title>Import projects</title>
  <para><mediaobject><imageobject>
    <imagedata fileref="images/egit-clone/09_import_projects_select.png
        " />
  </imageobject></mediaobject></para>
</step>
<step>
  <title>Switch to the Git Perspective</title>
  <para><mediaobject><imageobject>
    <imagedata fileref="images/egit-clone/10_git_switch_to_git.png" />
  </imageobject></mediaobject></para>
</step>
<step>
  <title>Switch to the Git Perspective</title>
  <para><mediaobject><imageobject>
    <imagedata fileref="images/egit-clone/11_git_switch_to_git.png" />
  </imageobject></mediaobject></para>
</step>
<step>
  <title>Enable Gerrit</title>
  <para><mediaobject><imageobject>
    <imagedata fileref="images/egit-clone/12_git_enable_gerrit.png" />
  </imageobject></mediaobject></para>
</step>
<step>
  <title>Configure Gerrit</title>
  <para><mediaobject><imageobject>
    <imagedata fileref="images/egit-clone/13_git_gerrit_config.png" />
  </imageobject></mediaobject></para>
  <para>Use <emphasis>saros-build.imp.fu-berlin.de</emphasis> and
  <emphasis>refs/for/master</emphasis></para>
</step>
<step>
  <title>Verify EGit being used in the Java Perspective</title>
  <para><mediaobject><imageobject>
    <imagedata fileref="images/egit-clone/14_switch_java_team.png" />
  </imageobject></mediaobject></para>
</step>
</procedure>
```

H Technische Probleme

Dies sind meine Vorschläge um technische Probleme in Saros zu lösen. Ich setze ein tieferes Verständnis voraus.

- Die beim Ausführen der STF Tests und von Benutzern hochgeladenen *Logfiles* sollten systematisch ausgewertet werden und die auftretenden Ausnahmen sollten beseitigt werden.
- Die *EditorManager* Klasse sollte in einen globalen Teil und einen Teil pro Sitzung geteilt werden.
- Elemente der grafischen Oberfläche, die sich auf eine aktive Sitzung beziehen, sollten dynamisch beim Starten der Sitzung erzeugt werden. Dies ermöglicht die Entfernung weiterer Klassen aus dem PicoContainer des `SarosContexts`.
- Die Modifikation von Aktivitäten in `ConcurrentDocumentClient` sollte durch einen generischen Filter geschehen.
- Die Benutzung von `instanceof` sollte durch das Erstellen von geeigneten Schnittstellen zurückgedrängt werden.
- Die `IActivityDataObject` Indirektion sollte entfernt werden.
- Auch im *readonly* Betrieb sollte der Jupiter Algorithmus verwendet werden. Das `TextEditActivityDataObject` und die Optimierungen im `ActivitySequencer` können dann gelöscht werden.
- Die Optimierung kleine Aktivitäten über IBB zu verschicken macht den `ActivitySequencer` komplizierter. Dies sollte durch einen Benchmark begründet werden.
- Das Verarbeiten und Erzeugen von Aktivitäten sollte immer aus dem gleichen Kontext geschehen. Viele der Klassen kommen nicht mit dem Verlassen von Benutzern an beliebigen Stellen klar. Ein Nutzer sollte nur nach dem Bearbeiten/Erzeugen aller Aktivitäten aus der Sitzung entfernt werden.
- Die Jenkins Konfiguration sollte die Gerrit Trigger auf einem anderen *Node* mit einem anderen Benutzer-Kontext ausführen. Damit könnte man allen Benutzern das Hochschieben von Änderungen erlauben.

Literatur

[1] R. Djemili, "Entwicklung einer Eclipse-Erweiterung zur Realisierung und Protokollierungverteilter Paarprogrammierung," Master's thesis, Freie Universität Berlin, Inst. für Informatik, 2006.

[2] S. Rossbach, "Einführung einer kontinuierlichen Integrationsumgebung und Verbesserung des Test-Frameworks," Nov. 2011.

[3] L. Chen, "Einführung eines Testprozesses," Master's thesis, Freie Universität Berlin, Inst. für Informatik, Apr. 2011.

[4] S. Szücs, "Behandlung von Netzwerk- und Sicherheitsaspekten in einem Werkzeug zur verteilten Paarprogrammierung," Master's thesis, Freie Universität Berlin, Inst. für Informatik, 2010.

[5] W. Belousow, "Verbesserung der Architektur und Dokumentation der DPP-Software Saros," Master's thesis, Freie Universität Berlin, Inst. für Informatik, July 2011.

[6] M. Johannsen, "Verbesserung und Pflege der Dokumentation der DPP-Software Saros," Master's thesis, Freie Universität Berlin, Inst. für Informatik, Oct. 2011.

[7] A. Waldmann, "Reviewboard comment to not be able to do deal with review feedback." http://saros-build.imp.fu-berlin.de/reviews/r/265/, Feb. 2012.

[8] A. Morton, "Tech Talk: Linus Torvalds on Git." https://git.wiki.kernel.org/index.php/LinusTalk200705Transcript, May 2007.

[9] L. Pionchon, "building html/pdf from Mallard." https://mail.gnome.org/archives/gnome-i18n/2011-May/msg00049.html, May 2011.