HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

INSTITUT
FÜR
SOFTWARE

Projektumgebung für Software Engineering Projekte

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2011

Autor(en):	Christian Zurbuchen, Beat Bodenmann
Betreuer & Experte:	Prof. Hans Rudin
Institut:	Institut für Software

Aufgabenstellung

Aufgabenstellung Studienarbeit für Herrn Beat Bodenmann und Herrn Christian Zurbuchen „Projektumgebung für Software Engineering Projekte"

1. Auftraggeber und Betreuer

Bei dieser Arbeit handelt es sich um eine HSR-interne Arbeit zur Unterstützung des Moduls Software-Engineering 2 sowie in zweiter Linie von Studien- und Bachelorarbeiten.

Betreuer HSR:

- Prof. Hans Rudin, (Verantw. Dozent, Auftraggeber und Betreuer HSR)
- Michael Gfeller, (Unterstützung Betreuung, Assistent IFS)
- Daniel Keller, (Beratung, Auftraggeber als Dozent Software Engineering)

2. Studierende

Diese Arbeit wird als Studienarbeit an der Abteilung Informatik durchgeführt von

- Beat Bodenmann
- Christian Zurbuchen,

3. Einführung

Das Modul „Software Engineering 2 – Projekt" muss ich Ihnen kaum vorstellen. Als Teilnehmer oder Teilnehmerin mussten Sie die Projektumgebung für Ihr SE2-Projekt selbst evaluieren und bereitstellen. Einzig ein paar Word- und Excel-Templates standen Ihnen zur Verfügung. Inzwischen gibt es eine Reihe interessanter Projektplattformen und Werkzeuge, die sich in den Projekten bewährt haben. Ziel dieser Arbeit ist das Bereitstellen einer Projektumgebung für SE2-Projekte, so dass künftige SE2-Projekteams nicht jedes Mal von Grund auf mit dem Aufbau einer Projektumgebung beginnen müssen und so rasch produktiv werden.

4. Aufgabenstellung

Das Bereitstellen einer SE2-Projektumgebung umfasst folgende Aufgaben:

- Festlegen der Anforderungen an eine solche Umgebung mit dem Betreuer .
- Evaluation einer geeigneten Plattform und einzelner Werkzeuge.
- Aufbau einer Projektumgebung mit evaluierten Tools und Integration von passenden Templates
- Bereitstellen eines Beispiel-Projektes in dieser Umgebung.
- Ausarbeiten von Templates, die in diese Projektumgebung passen.
- Deployment-Konzept für die Projekt- Umgebung (Wie kann man möglichst einfach eine SE2-Projektumgebung mit allen erforderlichen Templates bereitstellen?).

Die Aufgabenstellung ist offen. Sie können Ihre Erfahrung und Kreativität einbringen.

5. Zur Durchführung

Mit dem HSR-Betreuer finden wöchentliche Besprechungen statt. Alle Besprechungen sind von den Studenten mit einer Traktandenliste vorzubereiten, die Besprechung ist durch die Studierenden zu leiten und die Ergebnisse sind in einem Protokoll zu dokumentieren, das allen Beteiligten per E-Mail zugestellt wird.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen gemäss Projektplan sind einzelne Arbeitsresultate in vorläufigen Versionen abzugeben. Über die abgegebenen Arbeitsresultate erhalten die Studierenden ein vorläufiges Feedback. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Dokumentation.

6. Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen (siehe https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html). Die zu erstellenden Dokumente sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollten den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren. Die Dokumentation ist vollständig auf CD/DVD in 3 Exemplaren abzugeben. Auf Wunsch ist für den Auftraggeber eine gedruckte Version zu erstellen.

7. Termine

Siehe auch Terminplan auf https://www.hsr.ch/Termine-Diplom-Bachelor-und.5142.0.html.

Montag, 19. September 2011	Beginn der Studienarbeit, Ausgabe der Aufgabenstellung durch den Betreuer Erste kurze Besprechung mit Auftraggeber 12:00 bis 12:30 im 6.108
Dienstag, 27. September 2011	Terminreservation für Workshop mit Daniel Keller 15:10 bis 16:10 (Raum wird noch bekannt gegeben)
19. Dezember 2011	Die Studierenden senden folgende Dokumente der Arbeit per Email zur Prüfung an ihre Betreuer: Kurzfassung A0-Poster Vorlagen sowie eine ausführliche Anleitung betreffend Dokumentation stehen unter den allgemeinen Infos Diplom-, Bachelor- und Studienarbeiten zur Verfügung.
23. Dezember 2011	Die Studierenden senden die vom Betreuer abgenommene und freigegebene Kurzfassung als Word-Dokument an das Studiengangsekretariat (cfurrer(at)hsr.ch).
Freitag, 23. Dezember 2011	Abgabe des Berichtes an den Betreuer bis 17.00 Uhr.

8. Beurteilung

Eine erfolgreiche Studienarbeit zählt 8 ECTS-Punkte pro Studierenden. Für 1 ECTS Punkt ist eine Arbeitsleistung von ca. 25 bis 30 Stunden budgetiert. Für die Modulbeschreibung der Studienarbeit siehe auch https://unterricht.hsr.ch/Module/Modulinfo/Modulinfo.aspx .

Für die Beurteilung ist der HSR-Betreuer verantwortlich.

Gesichtspunkt	Gewicht
1. Organisation, Durchführung	1/5
2. Berichte (Abstract, Mgmt Summary, technischer u. persönliche Berichte) sowie Gliederung, Darstellung, Sprache der gesamten Dokumentation	1/5
3. Inhalt*)	3/5

*) Die Unterteilung und Gewichtung von 3. Inhalt wird im Laufe dieser Arbeit festgelegt.

Im Übrigen gelten die Bestimmungen der Abteilung Informatik für Studienarbeiten.

Rapperswil, den 17. September 2011

Prof. Hans Rudin
Institut für Software
Hochschule für Technik Rapperswil

Eigenständigkeitserklärung

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass wir sämtliche verwendeten Quellen erwähnt und korrekt angegeben haben.

Rapperswil, 22.12.2012

Christian Zurbuchen Beat Bodenmann

Projektumgebung für Software Engineering Projekte

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil
Herbstsemester 2011

Autoren: Christian Zurbuchen, Beat Bodenmann
Betreuer & Experte: Professor Hans Rudin

Datum: 22.12.2011

Inhalt

Abstract

Abteilung	Informatik
Namen der Studierenden	Christian Zurbuchen Beat Bodenmann
Studienjahr	Herbstsemester 2011
Titel der Studienarbeit	Projektumgebung für Software Engineering Projekte
Examinator	Professor Hans Rudin
Themengebiet	Software Engineering
Projektpartner	Institut für Software
Institut	Institut für Software

Das Ziel dieser Arbeit war es, den Studierenden eine State of the Art Projektumgebung zur Verfügung zu stellen, um ihnen einen raschen Einstieg ins Software Engineering 2 Projekt zu ermöglichen. Das beinhaltet eine Entwicklungsumgebung, Projektautomatisierung sowie Projektmanagement und Kollaborationsplattform. Dies wird unterstützt durch diverse Anleitungen und Tools. Das Ergebnis ist ein virtueller Server der alle Dokumente, Anleitungen, vorinstallierte Tools und ein Beispielprojekt enthält.

Nach einer sorgfältigen Evaluation einer Entwicklungsumgebung, verschiedenster Tools, wurde eine komplett einsatzfähige Projektumgebung für ein Java Projekt zusammengestellt. Um die Installation und das Einrichten dieser Tools zu vereinfachen wurden zu jedem empfohlenen oder verwendeten Tools Anleitungen mit Setup und Anwendungstipps erstellt.

Ein fertig aufgesetzter virtueller Server mit der Projektmanagementsoftware Redmine, einem Jenkins Buildserver und Git zur Versionskontrolle wird zur Verfügung gestellt.

Um ein Software Engineering 2 Projekt auch entsprechend dokumentieren zu können, werden Dokumentationsvorlagen vorgegeben. Es wurden die bereits vorhandenen Dokumente angepasst und erweitert, da der Fokus in diesem Projekt nicht in den einzelnen Dokumenten, sondern in der Veranschaulichung mit dem Code des dazu aufgesetzten Beispielprojektes lag.

Dieses Beispielprojekt wurde direkt auf der neu definierten Projektumgebung erstellt und deckt alle Softwareschichten ab. Auch alle Dokumente wurden dazu erstellt. Das Beispielprojekt wurde auf dem virtuellen Server ebenfalls in Redmine und Jenkins integriert, um den Zusammenhang zwischen den Ergebnissen und den Tools zu demonstrieren.

Technischer Bericht

1. Einführung

1.1 Problemstellung

Bei jedem Software Projekt muss das jeweilige Team zuerst eine Projektumgebung erstellen, bevor es mit der eigentlichen Arbeit beginnen kann. Dies ist auch in den Projekten im Rahmen des Unterrichtsmodul Software Engineering 2 Projekts der Fall.

Um künftigen Studenten den Einstieg in ein Projekt zu erleichtern wird diesen eine Projektumgebung bereitgestellt. Dies beinhaltet eine Entwicklungsumgebung, Projektautomatisierung sowie Projektmanagement und Kollaborationsplattform.

1.2 Vision

Bereitstellung einer funktionsfähigen Projektumgebung, Anleitungen zur Einrichtung und ein damit aufgebautes Beispielprojekt.

1.3 Aufgabenstellung

„Das Bereitstellen einer SE2-Projektumgebung umfasst folgende Aufgaben:

- Festlegen der Anforderungen an eine solche Umgebung mit dem Betreuer.
- Evaluation einer geeigneten Plattform und einzelner Werkzeuge.
- Aufbau einer Projektumgebung mit evaluierten Tools und Integration von passenden Templates
- Bereitstellen eines Beispielprojektes in dieser Umgebung.
- Ausarbeiten von Templates, die in diese Projektumgebung passen.
- Deployment-Konzept für die Projektumgebung (Wie kann man möglichst einfach eine SE2-Projektumgebung mit allen erforderlichen Templates bereitstellen?).“

Zitat 1: Technischer Bericht: Aufgabenstellung H. Rudin

1.4 Vorangegangene Arbeiten

Im Herbstsemester 2004/2005 erarbeiteten die damaligen Studenten Pascal Frey und Manfred Fäh zum einen diverse Dokumentationsvorlagen. Diese wurden im Umfang dieser Arbeit angepasst und erweitert.

1.5 Arbeitsumfeld

1.5.1 Rahmenbedingungen

Aufwand: Die Semesterarbeit wird mir 8 ECTS-Punkte honoriert. Dies bedeutet, gemäss der allgemeinen Definition von 30 Arbeitsstunden pro ECTS-Punkt, eine Arbeitsaufwand von 240 Stunden pro Student verteilt über 14 Wochen.

1.5.2 Infrastruktur

1.5.2.1 Hardware

Gerät	Beschreibung
2xHSR-Workstation	Von der HSR zur Verfügung gestellte Workstation mit Windows 7
HSR- vServer	Ubuntu 64bit Kernel-img 2.6.32-36-server
Externer Server	SVN Repository und Trac-Enviroment bei codemind.ch von David Schoettl
Acer Aspire 5820TG	Privater Laptop von Christian Zurbuchen mit Windows 7
Alienware mx15	Privater Laptop von Beat Bodenmann mit Windows 7

Tabelle 1: Technischer Bericht: Benutzte Hardware

1.5.2.2 Software

Tool	Beschreibung	Verwendung	Version
Trac	Webbasiertes Projektmanagement Tool	Projektmanagement	0.11.5
Eclipse	IDE	IDE und Evaluationsinstrument verwendet	3.7.1
TortoiseSVN	Repository -Verwaltungstool	Repository-Verwaltung	1.7.2
Putty	Communication Terminal	Verwaltung des vServer	0.61
Microsoft Office	Office-Paket	Erstellung der Dokumente	2007/2010
Redmine	Webbasiertes Projektmanagement Tool	Evaluation	1.2.1
Jenkins	Buildserver	Evaluation und Erstellung des Beispielprojekts	1.441
Diverse	Diverse Weiter Tools und Plug-Ins	Evaluation und Erstellung des Beispielprojekts	

Tabelle 2: Technischer Bericht: Benutzte Software (Auszug)

1.6 Vorgehensweise

Als Arbeitsprozessmodel wurde RUP gewählt, soweit dieses Model auf diese Art Arbeit angepasst werden konnte. Der Grund für diese Auswahl, ist die Aufteilung der Arbeit in einzelne Phasen und Iterationen mit jeweiligen Meilensteinen als Zwischenziele.

Die ersten zwei Wochen der Inception Phase wurden dazu genutzt um das Projekt zu initiieren und grob zu Planen. Des Weiteren wurden die Anforderungen festgelegt und erste Tools grob analysiert.

In den Wochen drei bis und mit fünf, der Elaboration Phase wurden Tools evaluiert und ausgetestet.

In der Construction Phase, welche sich über die Wochen 6 bis 12 ersteckte, wurde die Projektumgebung zusammengestellt, Anleitungen dazu verfasst und damit ein Beispielprojekt realisiert.

Die letzten 2 Wochen des Semester, die Transition Phase, wurden zur Überarbeitung der Dokumentation und dem Erstellen von Berichten genutzt.

Eine genauere Definition des Arbeitsablaufs kann dem Projektplan entnommen werden.

2. Resultate

2.1 Erreichte Ziele

Es ist uns gelungen eine einsatzfähige Projektumgebung zusammenzustellen, die nun von kommenden Studenten genutzt werden kann. Dank der auf dem virtuellen Server vorinstallierten und soweit wie möglich vorkonfigurierten Komponenten, Redmine, Jenkins und Git-Core, können die Studenten direkt mit ihrem Projekt beginnen ohne sich zu Beginn durch die meist nicht ganz trivialen Installationsroutinen kämpfen zu müssen.

Des Weiteren haben wir eine lokale Entwicklungsumgebung mit einigen nützlichen Plug-Ins zusammengestellt. Diesen können die Studenten, geführt durch verschiedene Anleitungen, auf ihrem persönlichen Rechner einrichten.

Nach Abschluss des Kernziels, der Projektumgebung, haben wir auf dieser ein kleines Beispielprojekt erstellt. Welches leider, aus Zeitgründen, etwas kleiner als geplant implementiert wurde.

2.2 Virtueller Server

Auf dem virtuellen Ubuntu Server wurden folgende Komponenten installiert und vorkonfiguriert:

- Redmine (Projektmanagementsoftware)
- Jenkins (Ein Buildserver zur kontinuierlichen Integration)
- Git-Core (Als Grundlage für ein GIT-Repository)

Die Entscheidungsgrundlage für diese Tools kann den entsprechenden Evaluationsdokumenten entnommen werden.

2.2.1 Startseite

Damit sich die Studenten nicht mit dem Konsolen – Interface des virtuellen Ubuntu Servers herumschlagen müssen, wurde ein via Webbrowser erreichbare Startseite eingerichtet. Auf dieser ist es möglich Redmine und Jenkins zu starten, sowie die Anleitungen zu allen Tools abzurufen. Auch die Dokumentenstruktur, welche als Vorlage verwendet werden kann, und die Ergebnisse des Beispielprojektes sind über diese Seite abrufbar.

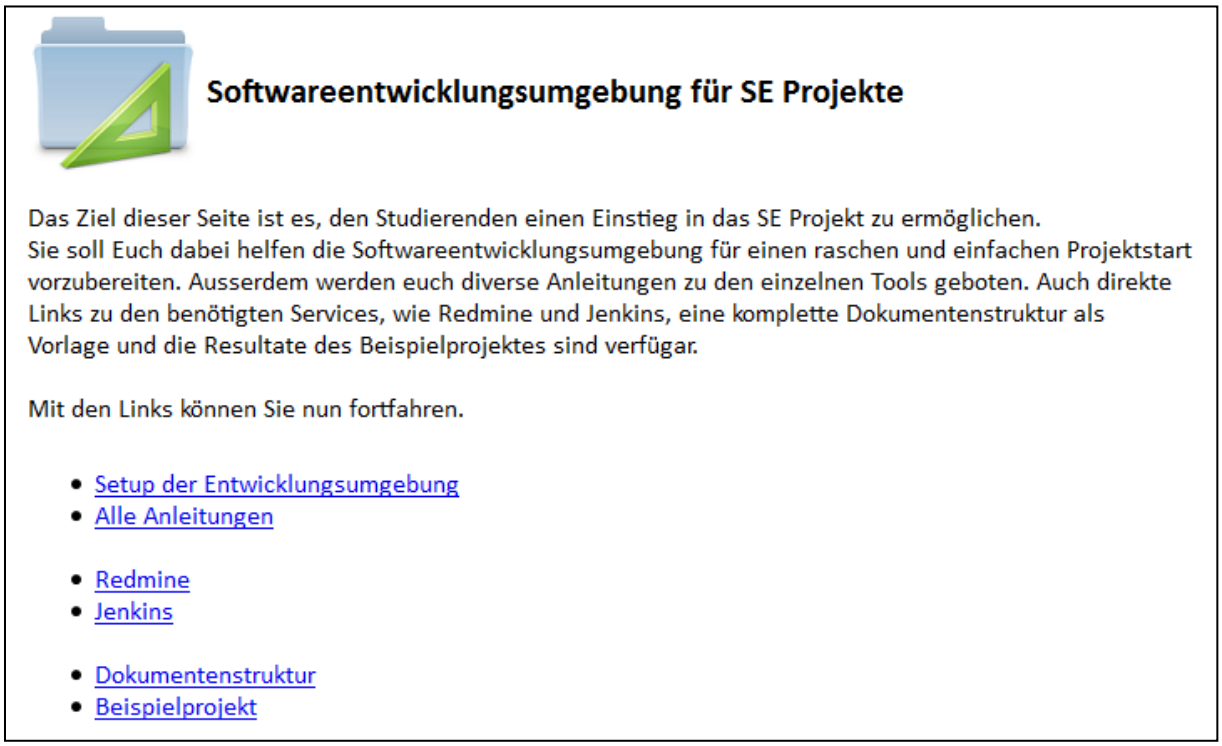


Abbildung 1: Technischer Bericht: Web-Startseite des Virtuellen Servers

2.2.2 Anleitung

Zur Hilfe bei der Einrichtung eines Projektes auf der Softwareentwicklungsumgebung wurde ein Wizard erstellt, der den Benutzer durch den Einrichtungsprozess führt. Die folgende Abbildung zeigt als Beispiel einen Teil der Anleitung zur Einrichtung des Versionsmanagement.

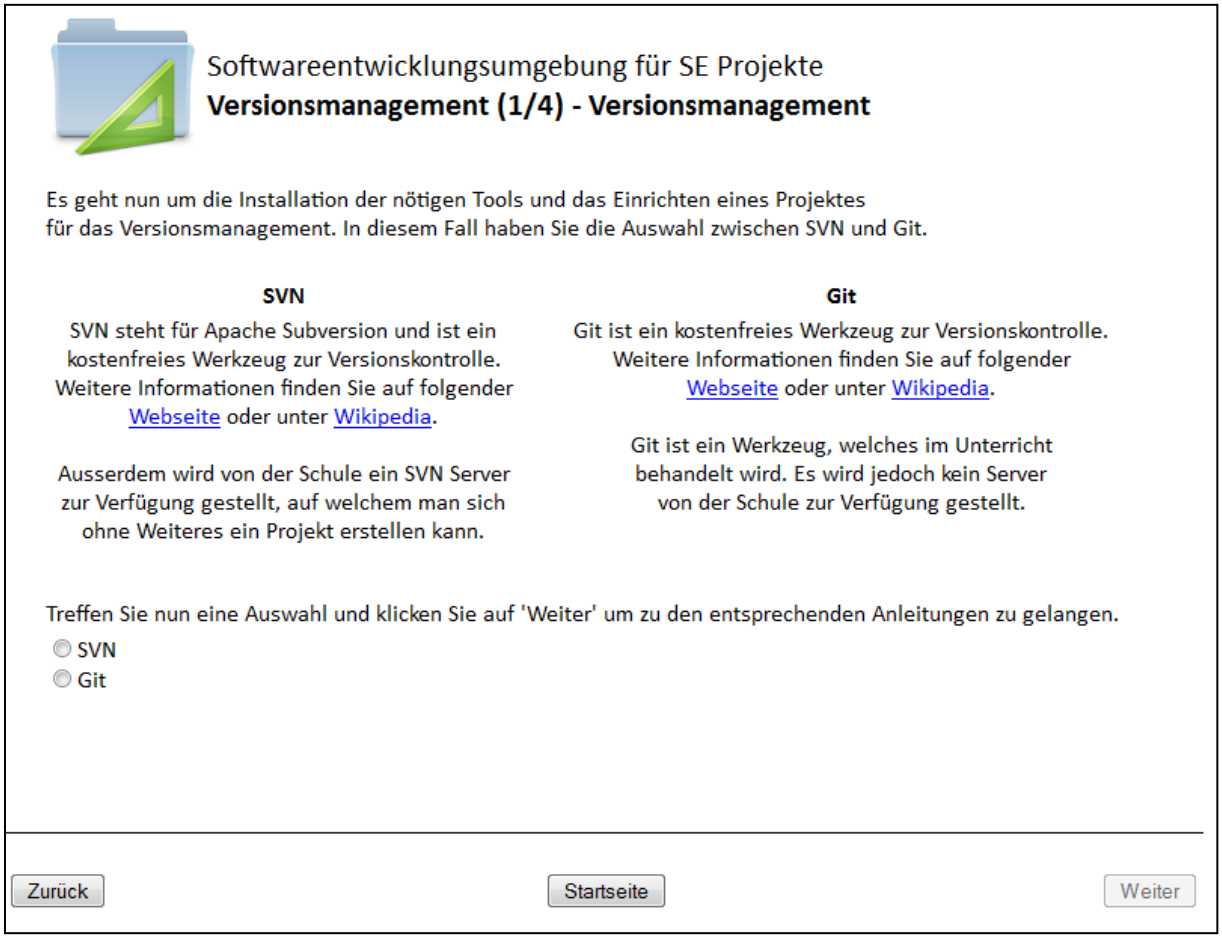


Abbildung 2: Technischer Bericht: Anleitung Wizard Virtueller Server

Auf den meisten Seiten, ist es dem Student möglich, eine Anleitung abzurufen, welche als PDF hinterlegt ist.

Die Anleitungen sind im Bereich Projektdokumente unter Anleitungen zu finden.

2.3 Lokale Entwicklungsumgebung

Als Grundlage haben wir uns für die Eclipse IDE entschieden und empfehlen folgende Erweiterungen:

- FindBugs (Überprüfung des Codes auf eventuelle Fehler)
- EclipseMetrics (Analyse des Codes bezüglich Qualität)
- STAN (Tool zur Strukturanalyse)
- CheckStyle (Überprüfung von Styleguidelines)
- WindowBuilderPro (Erstellung einfacher graphischer Oberflächen)

Die Entscheidungsgrundlage für diese Tools kann den entsprechenden Evaluationsdokumenten entnommen werden.

2.4 Beispielprojekt

Das Beispielprojekt, eine kleine lokale Java Applikation, wurde zur Veranschaulichung der Funktionalitäten der Projektumgebung erstellt. Ebenfalls wurden alle RUP Projektdokumente als konkretes Beispiel für die Studenten erstellt. Die Dokumente sind aufgrund der geringen Projektgrösse kleiner und weniger umfangreich als jene eines SE2 Projektes. Die Studenten werden jedoch an mehreren Stellen auf dies hingewiesen.

2.4.1 Umfang

Ursprünglich war eine Applikation zur Verwaltung der Absenzen ein Team geplant. Dies musste jedoch aus Zeitgründen auf eine reine Verwaltung der Mitglieder eines Teams reduziert werden.

2.4.1.1 Geplant

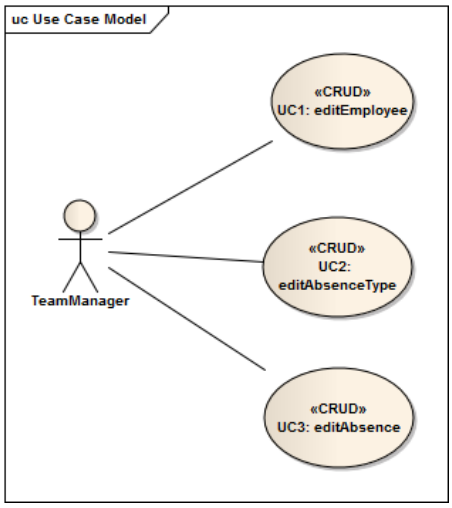


Abbildung 3: Technischer Bericht: Use Case Model Beispielprojekt

Ein Team Manager sollte eine Mitarbeiter erfassen und diesem eine Absenz von einem gewissen Typ hinzufügen können. Es handelt sich hierbei um reine CRUD Use Cases.

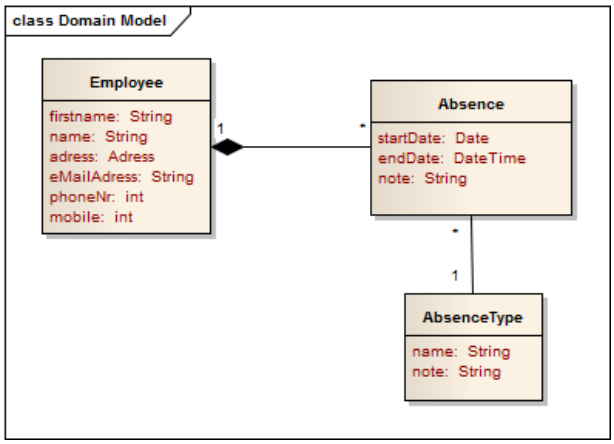


Abbildung 4: Technischer Bericht: Domain Model Beispielprojekt

Die Problemdomain wurde mit den drei Komponenten Employee, Absence und AbsenceType definiert.

2.4.1.2 Realisiert

Aus zeitlichen Gründen konnte nur eine Mitarbeiterverwaltung realisiert werden, da es wichtiger war ein lauffähiges und über alle Schichten operierendes Programm abzuliefern.

2.4.1.2.1 Packages

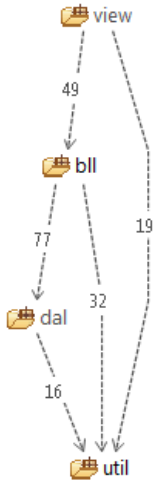


Abbildung 5: Technischer Bericht: Package Übersicht Beispielprojekt

Das view Package beinhaltet alle User Interface relevanten Klassen. Das Package ist in folgende weitere drei Packages unterteilt:

- **view**: Beinhaltet die Frames der Applikation.
- **view.element**: Beinhaltet die Elemente der Frames der Applikation.
- **view.draft**: Beinhaltet Vorlagen für die Elemente oder die Frames, welche verwendet werden.

Das bll Package (bll steht für Business Logic Layer) beinhaltet alle Klassen, welche mit der Business Logik im Zusammenhang stehen. Das Package ist in folgende weitere Packages unterteilt:

- **bll**: Enthält den Kontroller der Applikation, welcher die Schnittstelle zum Data Access Layer darstellt.
- **bll.domain**: Beinhaltet alle Domainklassen.
- **bll.domain.util**: Beinhaltet Hilfklassen, welche in der Business Logik verwendet werden.

Das dal Package (dal steht für Data Access Layer) beinhaltet alle Klassen, welche mit der Zugriff zur Datenhaltung im Zusammenhang stehen. Das Package ist in folgende weitere Packages unterteilt:

- **dal**: Enthält die Komponenten die den Zugriff auf die XML Datei gewährleisten und eine Mock Instanz, welche den Zugriff mockt.
- **dal.dto**: Beinhaltet die Data Transfer Objects (DTO), welche von der XML Datei direkt in eine Klassenstruktur gelesen werden.

Das util Package beinhaltet diverse Hilfsklassen, welche von den anderen Packages gemeinsam verwendet werden. Das Package ist in folgende weitere Packages unterteilt:

- **util**: Beinhaltet die beiden Hilfsklassen, welche den Zugriff zu den .properties Dateien gewährleisten.
- **util.exception**: Beinhaltet die applikationsspezifische Exception.

2.4.1.2.2 Code Qualität

Um die Code Qualität hoch zu halten, haben wir alle von uns empfohlenen Tools verwendet. Der Jenkins Buildserver sorgte für eine kontinuierliche Integration.

Wir haben speziell darauf geachtet, dass der grösste Teil unseres Codes getestet wird und wir eine hohe Coverage erhalten. Es ist jedoch anzumerken, dass das view Package nicht analysiert wurde.

Overall Coverage Summary

Name	Klassen	Methoden	Blöcke
all classes	88.2% 15/17	95.1% 117/123	94.1% 1387/1474

Coverage Breakdown by Package

Name	Klassen	Methoden	Blöcke
bll	100.0% 1/1	92.9% 13/14	84.9% 157/185
bll.domain	100.0% 3/3	100.0% 41/41	100.0% 507/507
bll.domain.util	100.0% 3/3	83.3% 15/18	97.0% 353/364
dal	100.0% 2/2	100.0% 7/7	84.3% 118/140
dal.dto	100.0% 3/3	100.0% 29/29	100.0% 144/144
util	50.0% 2/4	83.3% 10/12	79.2% 99/125
util.exception	100.0% 1/1	100.0% 2/2	100.0% 9/9

Abbildung 6: Technischer Bericht: Code Coverage Beispielprojekt

2.4.1.2.3 User Interface

Das User Interface wurde mit Swing realisiert und mit dem WindowBuilderPro erstellt.

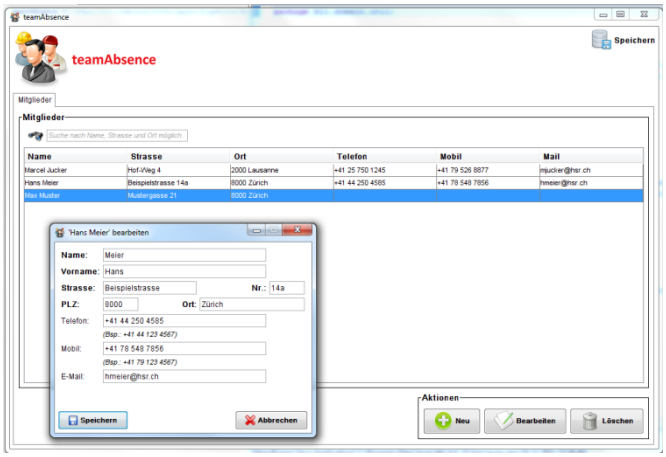


Abbildung 7: Technischer Bericht: User Interface Beispielprojekt

3. Persönliche Berichte

3.1 Beat Bodenmann

3.1.1 Projektverlauf

Zu Beginn war der schwierigste Punkt, die Erfassung eines klaren Ziels. Der in der zweiten Woche durchgeführte Anforderungsworkshop schaffte dann klare Verhältnisse, was von dieser Arbeit als Resultat erwartet wurde.

Die darauffolgende Evaluation der einzelnen Tools war für mich der interessanteste Teil der Arbeit. Durch die genaue Analyse der Tools konnte ich einige sehr nützliche Funktionen entdecken die ich zuvor nie genutzt hatte. Natürlich habe ich einige Tools, zum Beispiel STAN, bis anhin gar nicht gekannt und konnte sie so kennenlernen.

Jenkins zu evaluieren war sehr aufwendig und mit einigen Problemen verbunden, denn Apache-Ant zu verwenden ist auf Grund seiner Mächtigkeit nicht ganz einfach. Die Verbesserung meiner Ant-Kenntnisse hat deshalb auch viel Zeit in Anspruch genommen.

Als ebenfalls sehr lehrreich stellte sich die Einrichtung des virtuellen Servers heraus. Dadurch konnte ich meine Kenntnisse für Unix-Systeme massiv verbessern und erweitern.

Bei der Erstellung der Anleitungen für die Tools lag die Schwierigkeit darin, das wesentliche heraus zunehmen und dies verständlich darzustellen. Ich bin der Meinung dass dies uns sehr gut gelungen ist. Durch das gegenseitige Überprüfen der erarbeiteten Anleitungen im Team konnten wir sicherstellen, dass die Anleitungen hilfreich und vollständig sind.

Aufgrund unserer Bemühungen bei den Anleitungen, ist für das Beispielprojekt etwas weniger Zeit als anfänglich geplant. Wir sahen uns gezwungen den entsprechenden Meilenstein um eine Woche zu verschieben. Da das Projekt jedoch absichtlich so gewählt wurde, dass es sich beinahe beliebig erweitern und anpassen lässt, stellte dies kein Problem dar.

3.1.2 Rückblick

Das Projekt verlief soweit gut. Da ich und Christian bereits beim Software Engineering 2 Projekt zusammengearbeitet haben, wusste wir wie der Andere arbeitet und konnten uns darauf einstellen. Des Weiteren hat die klare Aufteilung der Verantwortlichkeit dazu geführt, dass wir beide stets produktiv arbeiten konnten und ermöglichte eine unabhängigere Qualitätskontrolle der Arbeit des Anderen.

3.1.3 Lessons Learned

Die Semesterarbeit war eine gute Übung im Hinblick auf die Bachelorarbeit. Man konnte die Herangehensweise an ein solches Projekt erlernen und vertiefen.

Ich habe einiges im Umgang mit der Ubuntu Konsole gelernt, sowie die Kenntnisse über Ant erweitert und viele neue Tools kennengelernt.

Als Kritikpunkt an unsere Arbeit würde ich die Zeiterfassung im Excel nennen, was jedoch nötig war, da wir als Projektmanagementsoftware Trac verwendeten, welche keine bauchbare Zeiterfassung ermöglicht. In meinem nächsten Projekt werde ich darauf bestehen Redmine zu verwenden.

3.2 Christian Zurbuchen

3.2.1 Projektverlauf

Am Anfang des Projektes war uns die Aufgabe noch nicht ganz klar. Wir wussten was uns inhaltlich erwarten würde, weshalb wir auch unser Interesse für diese Arbeit aussprachen, jedoch kannten wir den konkreten Lieferumfang nicht. Wie sich dann aber feststellte wurde der noch in den ersten zwei Wochen von uns und einem Expertenteam in einem Workshop erarbeitet. Die ersten zwei bis drei Wochen wurden also komplett für die Planung des Projektes und das korrekte Festlegen der Anforderung aufgewendet.

Als die Anforderungen spezifiziert waren und die grobe Planung des Projektes abgeschlossen wurde, konnten wir mit der eigentlichen Projektarbeit beginnen.

Wir begannen mit der Evaluation der einzelnen Tools. Dies beanspruchte sehr viel Zeit, da es in jedem Bereich der Softwareentwicklung diverse gute und sinnvolle Angebote gibt, welche wir alle einzeln überprüften um einen Favoriten in jedem Bereich auszumachen. Mir wurden auch sehr viele Mängel in vergangen Projekten, wie zum Beispiel das Software Engineering 2 Projekt, aufgezeigt während ich die einzelnen Tools evaluiert habe. Dies führt mich zum Entschluss in zukünftigen Projekten mehrheitlich solche Tools einzusetzen und anzuwenden.

Als nächster Punkt ging es darum den virtuellen Server mit Git, Jenkins und Redmine aufzusetzen. Diese Arbeit wurde grösstenteils von Beat erledigt, da er sich mit Servern besser auskennt. Die Arbeit war aufwendiger als gedacht und bereitete noch so manche kleine Probleme, welche schlussendlich jedoch alle behoben werden konnten. Während dieser Zeit wurden auch diverse Anleitungen zu den einzelnen Tools ausgearbeitet. Dies war für mich sehr interessant, weil ich so auch für mich persönlich herausfinden konnte, wie ich in Zukunft die einzelnen Tools sinnvoll einsetzen kann. Auch wurde die Startseite des virtuellen Servers um eine HTML Seite ergänzt, welche die ganzen Anleitungen und Ergebnisse unseres Projektes beinhaltet. Diese Aufgabe floss mehrheitlich in meinen Bereich, da ich mich mit HTML besser auskenne.

Nun da der virtuelle Server aufgesetzt ist und die meisten Anleitungen fertiggestellt wurden, setzten wir ein Beispielprojekt auf der neu geschaffenen Projektumgebung auf. Dies war extrem hilfreich um nochmals alle Anleitungen und das korrekte Setup des virtuellen Servers zu überprüfen. Bei dieser Arbeit kamen auch noch diverse kleine Mängel und Fehler zum Vorschein. Die Überprüfung mit diesem Beispielprojekt fand ich persönlich extrem hilfreich und sie zeigte auch entsprechende Problembereiche in den Anleitungen auf. Leider hatten wir für das Beispielprojekt nicht genügend Zeit eingeplant, was uns dazu veranlasste den entsprechenden Meilenstein um eine Woche zu verschieben. Auch damit konnte nicht die komplett geplante Applikation implementiert werden, was jedoch keine schlimmen Folgen hatte. Das Ziel mit dem Source Code war lediglich zu zeigen wie dieser korrekt aufgebaut werden kann und um eine saubere Testabdeckung zu demonstrieren, was auch mit einer kleineren Problemdomain möglich war.

3.2.2 Rückblick

Da wir bereits im Software Engineering 2 Projekt zusammengearbeitet haben wurde eine sinnvolle Aufteilung der einzelnen Verantwortlichkeiten stark vereinfacht. Man kennt sich bereits und weiss auch, in welchen Bereichen der Andere einen guten Beitrag leisten kann und in welchen eher man selbst die Verantwortung übernehmen soll. Durch diese Aufteilung kamen wir auch mit der Einhaltung des Zeitplans gut zurecht und hatten oftmals noch genügend Zeit um die Arbeiten des anderen Teammitgliedes nochmals zu kontrollieren.

3.2.3 Lessons Learned

Ich finde eine bessere Vorbereitung auf eine erfolgreiche Bachelorarbeit gibt es nicht. Die Semesterarbeit bietet einem die Möglichkeit all die gesammelten Erfahrungen aus dem Software Engineering 2 Projekt anzuwenden und die neu gemachten direkt in die Bachelorarbeit einfliessen zu lassen.

Die Aufbereitung einer Projektumgebung hat mir persönlich sehr geholfen, da ich auch geschäftlich an Software Projekten arbeite. Ich kenne nun genügend Tools und Plug-Ins um ein Software Projekt gut zu planen, koordinieren und im Endeffekt auch erfolgreich durchzuführen. Interessant war auch, dass ich für einige Tools die ich seit Jahren verwende bei der Evaluierung bessere Tools fand, die ich in Zukunft nun auch an deren Stelle anwenden werde.

Wir haben unser Projekt mit Trac und einer Zeiterfassung begonnen und später bei der Evaluation einer Projektmanagementsoftware sind wir auf Redmine gestossen, welches diese beiden Funktionalitäten sauber und übersichtlich vereint. In Zukunft werde ich bei solchen Projekten nur noch mit Redmine arbeiten, sofern ein dafür benötigter, Server zur Verfügung steht.

Was ich ebenfalls als Erfahrung mitnehme in zukünftige Projekte ist das Setzen von Terminen bei einzelnen Arbeitspaketen. Teilweise wurde eine Aufgabe relativ knapp vor dem Abgabetermin erledigt, was es dem anderen Teammitglied nicht einfach machte seine Ergänzungen oder Korrekturen noch anzufügen. Ich möchte in Zukunft in Projekten immer klare Termine für einzelne Arbeitspakete setzen. So hat jeder im Team die Möglichkeit die abgeschlossene Arbeit nochmals zu überprüfen.

Bei der Entwicklung der User Interfaces für das Beispielprojekt stellte ich fest, das ein Testen dieser nicht ohne weiteres möglich ist. Beim nächsten Projekt mit Swing möchte ich die entsprechende Logik der User Interfaces in eigene Klassen extrahieren um diese sauber testen zu können.

Projektdokumente

4. Projektplan

4.1 Einführung

4.1.1 Zweck

Dieses Dokument soll einen Überblick über die Planung und die Durchführung des Projekts SEPEnv (Softwareentwicklungsumgebung für SE Projekte) verschaffen.

4.1.2 Gültigkeitsbereich

Die Gültigkeit dieses Projektplans erstreckt sich über das Herbstsemester 2011 im Rahmen einer Semesterarbeit.

4.1.3 Referenzen

Nachfolgend werden die Dokumente aufgeführt, welche von diesem Dokument referenziert werden.

- Aufgabenstellung.pdf (vom Beteuer)
- Anforderungsspezifikation.pdf
- Glossar.pdf

4.1.4 Übersicht

Im folgenden Abschnitt wird eine Übersicht über das Projekt SEPEnv gewährt. Es soll ersichtlich werden, was der Zweck und die Ziele des Projektes sind.

Später werden die einzelnen Mitarbeiter des Projekts mit ihren Aufgaben und Verantwortungen beschrieben.

Ein wichtiger Teil dieses Dokumentes ist der Abschnitt ‚Zeitliche Planung'. Es werden dort der Projektplan, der Zeitplan, die Iterationsplanung und die Meilensteine festgelegt.

Des Weiteren erörtern wir die möglichen Risiken in unserem Projekt und schätzen den möglichen Schaden ein.

4.2 Projekt Übersicht

4.2.1 Zweck und Ziele

Das Ziel dieser Arbeit ist, den Studenten eine Entwicklungsumgebung zur Verfügung zu stellen, um ihnen einen raschen Einstieg ins Software Engineering 2 Projekt zu ermöglichen.

Die Arbeit umfasst die Bereitstellung von Dokumentationsvorlagen, Evaluierung von verschiedenen Tools und Software, Einrichten eines Images für einen Virtuellen Server und das Aufsetzen eines Beispielprojektes auf der neu bereitgestellten Umgebung. Der Student soll eine Art direkten Start ins Projekt bekommen unterstützt mit Anleitungen. Er soll wenn er mit der Entwicklung anfängt wissen, was die Anforderungen an ein solches Projekt sind mit dem Fokus auf die Entwicklung des Codes.

Das Projekt an sich dient nach dem Software Engineering 2 Projekt erneut zur Vertiefung der Lehrstoffe und als Vorbereitung auf die Bachelorarbeit.

4.2.2 Lieferumfang

Folgende Punkte sind im Lieferumfang enthalten:

- Ein Server Image mit dem Beispielprojekt und den relevanten Tools bereits laufend.
- Anleitung (im HTML-Format oder ähnliches), welches die Einrichtung der Projekte (z.B. im Versionsmanagement) erleichtert.
- Dokumentenstruktur mit allen relevanten Dokumenten, welche auf dem Server Image ebenfalls integriert ist.
- Zusätzliche Anleitungen/Empfehlungen im Format PDF ebenfalls auf dem Server Image in der Dokumentenstruktur.

Weitere Details zum Lieferumfang befinden sich noch im Dokument *Anforderungen.pdf*.

Definierte Abgaben für die Semesterarbeit können den Anleitungen auf hsr.ch entnommen werden.

4.2.3 Annahmen und Einschränkungen

Die einzelnen Projektmitarbeiter werden gemäss ihren Vorkenntnissen laufend den neu definierten Arbeitspaketen zugeteilt. Wir rechnen mit einem Arbeitsaufwand von 17 Stunden (30 * 8 = 240 / 14) pro Woche, wovon 1 Stunde für regelmässige Sitzungen mit dem Betreuer und 1 Stunde für interne Sitzungen, welche nur Projektmitarbeiter besuchen, verwendet werden.

Der Fokus des Projektes liegt klar bei der Programmiersprache Java. Falls gegen Ende noch genügend Zeit vorhanden ist, werden noch Anleitungen / Empfehlungen für C# gemacht.

4.3 Projektorganisation

4.3.1 Organisationsstruktur

Name	Aufgaben	Verantwortung
Christian Zurbuchen	- Überwachung Projektplan und Zeiterfassung - Sitzungsvorbereitung - Entwicklung Beispielprojekt - Analyse von Software Tools - Überwachung der Qualität	- Überwachung des Zeitplans - Planung des Projektes (Arbeitspakete zu den einzelnen Iterationen/Phasen) - Koordination des Projektes / Teams
Beat Bodenmann	- Überwachung gesamte Projektdokumentation - Setup und Überwachung virtueller Server - Analyse von Projektmanagement Tools (Redmine, Trac, usw.) - Überwachung der Qualität	- Protokolle erstellen - Überwachung der Qualität der Dokumente - Management des virtuellen Servers

Tabelle 3: Projektplan: Organisationsstruktur

4.3.2 Externe Schnittstellen

Betreuer	Hans Rudin
	Daniel Keller
	Michael Gfeller

Tabelle 4: Projektplan: Externe Schnittstellen

4.4 Managementabläufe

4.4.1 Kostenvoranschlag

Bis zu 14 Wochen sind geplant für das Projekt. Nach 12 Wochen sind die Hauptaufgaben jedoch beendet und in den letzten zwei Wochen wird der Projektabschluss vorbereitet und letzte Abschlussarbeiten werden noch erledigt.

4.4.2 Zeitliche Planung

In diesem Abschnitt wird die zeitliche Planung des Projektes gemacht. Die Planung und Einteilung der Phasen und Iterationen, sowie die Festlegung der einzelnen Meilensteine.

Die folgende Abbildung soll einen Überblick über die Phasen / Iterationen geben. Die Meilensteine sind als rote Punkte direkt in der Planung markiert.

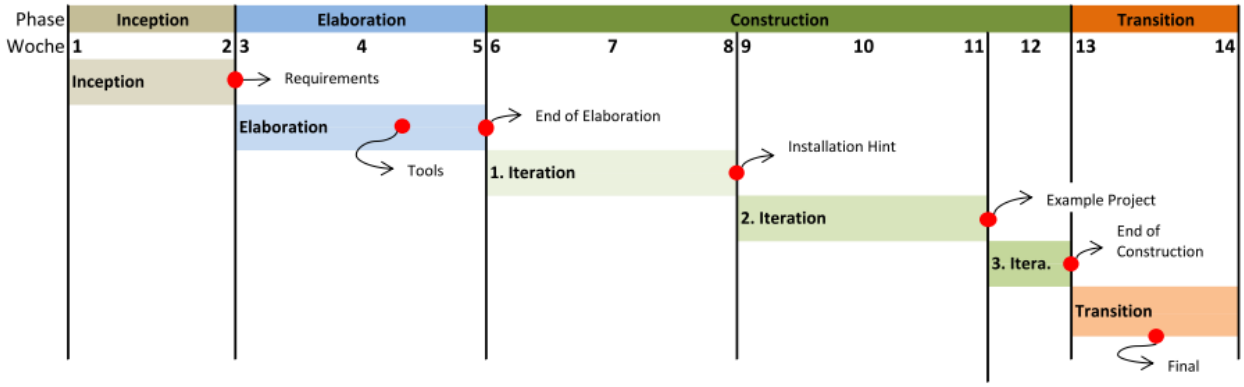


Abbildung 8: Projektplan: Überblick Phasen / Iterationen

4.4.2.1 Phasen / Iterationen

Das Projekt dauert total 14 Wochen. Diese 14 Wochen werden auf die in den folgenden Abschnitten beschriebenen Phasen aufgeteilt.

4.4.2.1.1 Inception (2 Wo)

In der Phase Inception geht es darum das Projekt zu initiieren und die Anforderungen festzulegen. Auch wird die Planung der Phasen / Meilensteine ausgearbeitet. Erste Quellen werden gesucht und auch schon erste Tools werden grob analysiert.

4.4.2.1.2 Elaboration (3 Wo)

Die Elaboration Phase dient zur Evaluierung und Auswertung der zu verwendenden Tools. Die verschiedenen Varianten werden ausgetestet und bewertet um eine für die Schule möglichst sinnvolle Variante zu finden.

4.4.2.1.3 Construction (7 Wo)

Die Construction Phase wird in drei Iterationen aufgeteilt, diese werden in den folgenden Abschnitten beschrieben.

4.4.2.1.4 Iteration 1 (3 Wo)

Alle erarbeiteten Ergebnisse aus der Elaboration Phase werden nun aufbereitet um sie für die Studenten anwendbar zu machen. Server werden aufgesetzt, Anleitungen ausgearbeitet, Dokumente werden bereitgestellt und eine Installationshilfe wird erstellt.

4.4.2.1.5 Iteration 2 (3 Wo)

Ein Beispielprojekt wird auf der neuen Umgebung, welche in der Construction Phase bereitgestellt wurde, aufgesetzt. Wichtig bei dem Projekt sind ein vollständiges vorgehen nach RUP sowie sauberes Code Design, Testabdeckung und automatische Building-Prozesse. Der Student sollte sich möglichst rasch im Code und der Dokumentation zu Recht finden können und schnell wissen, worauf er bei der Erstellung achten muss.

4.4.2.1.6 Iteration 3 (1 Wo)

Mängel und Probleme, welche in der zweiten Iteration festgestellt wurden werden in der abschliessenden Iteration der Construction Phase behoben. Falls grössere Probleme auftreten wird dies mit dem Betreuer besprochen, in welchem Umfang wir dies noch berücksichtigen können in der Semesterarbeit.

4.4.2.1.7 Transition (2 Wo)

Alle Resultate werden für die Abgabe aufbereitet. Letzte Abschlussarbeiten werden allenfalls noch erledigt. Die Berichte, Dokumentation usw. werden aufbereitet und abgeschlossen.

4.4.2.2 Meilensteine

In den folgenden Abschnitten sind die Meilensteine des Projektes beschrieben.

4.4.2.2.1 Requirements

Termin: 2. Oktober 2011

Alle Anforderungen an das Projekt sind klar definiert und ausgearbeitet. Ebenfalls sind sie vom Betreuer angenommen.

4.4.2.2.2 Tools

Termin: 16. Oktober 2011

Bei der Erreichung dieses Meilensteins sind die benötigten Tools ausgewählt. Die Tools wurden evaluiert und bewertet und wurden nach relevanten Bedingungen geprüft. Die Tools sind früher eingeplant, da es schwer abzuschätzen ist, wie viel Zeit die Analyse in Anspruch nehmen wird.

4.4.2.2.3 End of Elaboration

Termin: 23. Oktober 2011

Zum Ende der Phase Elaboration wird dieser Meilenstein erreicht. Alle Tools, Dokumente und weitere Vorschläge für SE Projekte sind ausgewählt.

4.4.2.2.4 Installation Hint

Termin: 13. November 2011

Die Anleitung und Durchführung zum Setup eines Projektes ist in groben Zügen erledigt. Der Student kann sich durch eine Reihe von Hinweisen und Tipps klicken und so die nötige Software und die Tools aufsetzen und installieren. Erste Anleitungen sind ebenfalls schon erstellt.

4.4.2.2.5 Example Project

Termin: 4. Dezember 2011

Ein Beispielprojekt ist von A bis Z auf der neu aufgesetzten Projektumgebung aufgesetzt. Die vorgeschlagenen Tools und Dokumente sind ebenfalls verwendet worden.

4.4.2.2.6 End of Construction

Termin: 11. Dezember 2011

Alle Anleitungen, Hinweise und Tipps sind fertiggestellt. Die Dokumente sind vorbereitet und die Entwicklungsumgebung ist aufbereitet. Alles ist also bereit für die Entwicklung eines Beispielprojektes.

4.4.2.2.7 Final

Termin: 18. Dezember 2011

Alles ist vorbereitet für die Abgaben und das Projekt ist grösstenteils abgeschlossen.

4.4.3 Besprechungen

Das Projektteam trifft sich wöchentlich für eine Stunde um Probleme zu besprechen, das weitere Vorgehen zu planen und Erledigtes zusammenzuführen. Dies findet normalerweise jeweils Montag von 13 bis 14 Uhr statt.

Vor Abschluss eines Meilensteins werden noch zusätzliche Besprechungen einberufen um etwaige Probleme oder noch ausstehende Arbeiten zu klären und zu verteilen.

Die Besprechungen mit dem Betreuer finden ebenfalls wöchentlich statt, jeweils Montag um 15 Uhr. Sie dauern in der Regel etwa eine Stunde.

4.5 Risikomanagement

4.5.1 Risiken

Im Folgenden sind Massnahmen beschrieben die beim Eintreten von Risiken angewendet werden oder Massnahmen um Risiken zu vermeiden.

Allgemeine Risiken:

- Bei Problemen mit Hardware oder Software wird versucht möglichst schnell Ersatz zu finden. Beispielweise andere Workstations.
- Datenverlust wird von Anfang an soweit verhindert, dass im schlimmsten Fall ein Arbeitstag verloren geht. Dank Subversion sind die Daten auf dem Server und auf allen Arbeitsstationen gesichert.
- Betrifft es den Ausfall eines Teammitglieds, wird eine Sitzung einberufen und um die Aufgaben neu zu gliedern und allenfalls Teile zu streichen.
- Den fehlenden Fachkenntnissen und der Komplexität wird durch Technologiestudium vorgebeugt.

Projektspezifische Risiken:

Wie dem Dokument entnommen werden kann gibt es nicht sehr viele projektspezifische Risiken. Dass liegt daran, da wenig bis kein Umgang mit neuen Technologien in unserem Projekt vorkommt. Viele unserer Risiken können leicht im Vorfeld schon vorgebeugt werden, es handelt sich mich um Dinge die abgeklärt werden müssen.

- Bei Problemen mit dem Server wird versucht möglichst schnell entweder einen Ersatz zu finden oder das Problem zu beheben.
- Essentielle Bereiche bei Tools werden so früh wie möglich geprüft, damit nicht später ein böses Erwachen folgt. Im Dokument *Anforderungen.pdf* sind die einzelnen essentiellen Anforderungen an ein Tool genau beschrieben. Bei einer Evaluation sollten also zuerst diese Anforderungen überprüft werden.
- Da in unserem Projekt die Anforderung schwer zu definieren sind und vieles möglich ist wird zu Beginn viel Wert auf eine detaillierte Definierung gelegt, damit später keine Probleme auftreten.
- Es kann eine zu grosse Auswahl an Tools geben, deshalb sollte früh auf Bewertungen und Erfahrungen von Anderen eingegangen werden um sich auf die besten Tools konzentrieren zu können und nicht den Überblick zu verlieren.

Grundsätzlich werden bei Problemen immer alle informiert, damit eventuell auch gemeinsam eine Lösung gefunden werden kann.

4.6 Qualitätsmassnahmen

Folgende Tabelle gibt einen Überblick über die einzelnen Qualitätsmassnahmen und deren Ziele. Einzelne Massnahmen sind während dem ganzen Projekt relevant, andere nur in bestimmten Phasen.

Massnahme	Zeitraum	Ziel
Fortlaufende Zeiterfassung	ganzes Projekt	ständiger Ist / Soll abgleich mit den erfassten Arbeitspaketen
Reviews im ganzen Team (Statusberichte, Risiko-Evaluierung, Planung, Protokoll)	mind. jede Woche einmal	ständige Überwachung des Projektstatus
Detaillierte Beschreibung der Anforderungen.	Inception / Elaboration	Eine klare Beschreibung aller Anforderungen im Detail um später auf keine Probleme zu stossen. Dies ist besonders wichtig, da die Anforderungen zu Beginn nicht sehr klar definiert sind und ein grosser Freiraum besteht.
Vor der Analyse der Tools Bewertungen/Erfahrungen einbeziehen.	Elaboration / Construction	Da es teilweise eine grosse Auswahl an Tools gibt sollte vorgängig auf Bewertungen und Erfahrungen anderer eingegangen werden um die Analyse auf die besten Tools einzugrenzen.
PM und SE Techniken gemäss Software Engineering 1 & 2 anwenden.	ganzes Projekt	Aus Erfahrungen anderer lernen und damit effizienter arbeiten. Qualitätsniveau erhöhen.

Tabelle 5: Projektplan: Qualitätsmassnahmen

4.6.1 Dokumentation & Projektmanagement

Alle Dokumente sind immer für alle Teammitglieder erreichbar und auf einem SVN Server abgelegt. Damit können die Dokumente fortlaufend Quergelesen werden. Für jede Abgabe wird jemand verantwortlich gezeichnet die erforderlichen Dokumente für die Abgabe zusammenzustellen.

Die Zeiterfassungstabellen sind ebenfalls auf dem SVN Server abgelegt. Jedes Teammitglied ist für seine Zeiterfassung selbst verantwortlich.

4.6.1.1 Trac Environment

Das Trac Environment für das Projekt befindet sich unter nachfolgender URL. Die Umgebung ist passwortgeschützt; Logins mit entsprechenden Zugriffsrechten wurden an die Team-Mitglieder abgegeben.

URL: https://svn.codemind.ch/trac/hsr/SEPEnv/

Login: guest / Passwort: guest

4.6.1.2 Trac Einsatz

Trac dient für uns zur Erfassung der Meilensteine und zur Iterationsplanung. Wird eine Iteration geplant so werden in Trac dafür die entsprechenden Arbeitspakete erfasst und den Personen zugeteilt.

4.6.2 Testen

In unserem Projekt wird kein Produkt (Software) als solches entwickelt, deshalb können nicht spezifische Tests geplant werden.

Unser Beispielprojekt, welches wir mit Hilfe unseres Images und den dazugehörigen Anleitungen erstellen werden, soll als ein grosser Test dienen um Mängel in unserer Lösung aufzuzeigen und diese auszumerzen. Jedes Projektmitglied soll sich möglichst mit allen Bereichen des Setups vertraut machen um die Effizienz der Lösung einschätzen zu können.

Um die Auswahl der am besten geeigneten Tools zu garantieren werden diese evaluiert und einander gegenübergestellt. So kann eine sinnvolle Auswahl gewährleistet werden und beim Setup des Beispielprojektes sollten nicht mehr allzu viele Mängel auftauchen.

5. Anforderungsspezifikation

5.1 Einführung

5.1.1 Zweck

Dieses Dokument dient zur Beschreibung aller Anforderungen an eine Softwareentwicklungs-umgebung basierend auf Java für zukünftig durchzuführende SE Projekte.

5.1.2 Beschreibung

Das Ziel dieser Arbeit ist, den Studenten eine Entwicklungsumgebung zur Verfügung zu stellen, um ihnen einen raschen Einstieg ins Software Engineering 2 Projekt zu ermöglichen.

Die Arbeit umfasst die Bereitstellung von Dokumentationsvorlagen, Evaluierung von verschiedenen Tools und Software, Einrichten eines Images für einen Virtuellen Server und das Aufsetzen eines Beispielprojektes auf der neu bereitgestellten Umgebung. Der Student soll eine Art direkten Start ins Projekt bekommen unterstützt mit Anleitungen. Er soll wenn er mit der Entwicklung anfängt wissen, was die Anforderungen an ein solches Projekt sind mit dem Fokus auf die Entwicklung des Codes.

5.1.3 Quellen

Jede Anforderung wurde mit einer Quelle versehen, woher sie stammt. Folgendes sind die möglichen Quellen:

- **Aufgabenstellung**: Punkte, die allein aus der Aufgabenstellung hervorkommen.
- **Workshop**: Wurden am Workshop besprochen und festgehalten (Projektteam und Betreuer).
- **Projektteam**: Ideen, welche vom Projektteam stammen.
- **spätere Reviews**: Bereiche, welche erst in späteren Reviews / Meetings entdeckt wurden.

5.2 Dokumente

Dem Studenten wird eine Ordnerstruktur mit den verlangten Dokumenten angeboten. Der Fokus in diesem Projekt liegt zwar nicht bei diesen Dokumenten, es werden aber trotzdem saubere und zeitgemässe Dokumente zur Verfügung gestellt.

Die wichtigen Bereiche werden in den folgenden Kapiteln beschrieben.

5.2.1 Projektantrag (Vision)

Quelle: Workshop

Das Dokument sollte die konkrete Vision des Projektes beschreiben und dem Auftraggeber einen Eindruck machen können um in von der Idee überzeugen zu können.

5.2.2 Projektplan

Quelle: Workshop

Der Projektplan in Zukunft sollte nicht mehr die genaue Auflistung der Arbeitspakete beinhalten. Dies ist zu Beginn des Projektes schwer zu planen und einzuschätzen.

Dieses Dokument soll die Liefergegenstände, Phasen und die Meilensteine des Projektes im Groben beschreiben. Auch eine Analyse der technischen Risiken soll in dieser Phase des Projektes gemacht werden. Ein Kapitel über qualitätssichernde Massnahmen ebenfalls.

Ein Plan wie die Tests gemacht werden ist unter den qualitätssichernden Massnahmen definiert.

5.2.3 Anforderungsspezifikation

Quelle: Workshop

Die Anforderungsspezifikation soll alle Anforderungen an das Produkt genau beschreiben. Es soll dem Team ermöglichen das Ziel des Projektes genauer zu definieren.

5.2.4 Domainanalyse

Quelle: spätere Reviews

Die Domainanalyse findet in einem eigenen Dokument seinen Platz.

5.2.5 Software Architecture Document (SAD)

Quelle: Workshop

Das SAD wird gemäss Vorlagen angeboten. Dieses Dokument beschreibt das Design und die Architektur des Produktes. Mögliche Prototypen werden ebenfalls beschrieben, falls vorhanden.

5.2.6 Sitzungsprotokolle

Quelle: Workshop

Eine einfache anpassbare Vorlage für ein Sitzungsprotokoll wird angeboten. Damit es den Studenten leicht fällt dies zu ergänzen und zu sehen welche Punkte wichtig sind und wozu so ein Protokoll auch dienen kann.

Falls möglich könnte dies direkt in einem Projektmanagement Tool integriert werden und so könnten direkt neue Arbeitspakete oder Bugs definiert werden.

5.2.7 Iterationsplanung und –assesment

Quelle: Workshop

Zu Beginn jeder Iteration (Phase) wird eine Planung dieser Iteration gemacht. Diese beinhaltet die Erfassung der neuen Arbeitspakete und die grobe Arbeitsaufteilung. Es kann direkt in einem Projektmanagement Tool integriert werden und sollte aber exportfähig sein um dem Dozenten einen Überblick zu ermöglichen.

Das Iterationsassesment beinhaltet die Analyse der vergangen Iteration um mögliche Fehler bei der vergangen gemachten Planung zu lokalisieren und versuchen diese in Zukunft zu vermeiden. Dieses Assesment wird am Ende einer Iteration durchgeführt und logischerweise vor der Planung der nächsten. Die erhaltenen Ergebnisse aus dem Assesment werden in einem Dokument festgehalten. Diese Vorlage wird den Studenten zur Verfügung gestellt.

5.2.8 Qualitätsmanagement

Quelle: Workshop

Massnahmen zur Qualitätssicherung werden hier vorgeschlagen und definiert. Reviewdokumente oder Ähnliches werden hier angeboten.

Auch der Bereich des Testens findet hier seinen Platz in Form von Testspezifikationen mit Protokollen und Resultaten.

5.2.9 Schlusspräsentation

Quelle: Workshop

Es geht hier darum den Studenten zu zeigen, was alles Teil einer solchen Präsentation sein sollte und wie man diese aufbauen könnte. Es werden wiederum Empfehlungen abgegeben und möglicherweise eine Art Checkliste was zur Abgabe noch alles beachtet werden muss.

5.2.10 Schlussbericht

Quelle: Workshop

Ein Schlussbericht mit der Zielerreichung und allgemeine Erfahrungen. Auch die Erfahrungsberichte der einzelnen Teammitglieder sind hier definiert (ein Abschnitt pro Teammitglied).

5.3 Entwicklung

Eine HTML-Seite mit Tools Empfehlungen, deren Beschreibung (Installationshinweise, Guide) und dem Link zur Download-Seite wird den künftigen Studenten zur Verfügung gestellt. Des weiter wird ein V-Server Image bereitgestellt, auf welchem unteranderem eine Projektmanagement Software sowie ein Build-Server laufen sollte.

Der Fokus wird bei der Entwicklung auf Java gesetzt und die Umgebung wird auch auf Java aufgesetzt. Falls noch genügend Zeit ist werden für C# noch Empfehlungen abgegeben und Anleitungen zur Verfügung gestellt.

Die wichtigen Punkte zur Unterstützung der Software Entwicklung werden in den folgenden Kapiteln beschrieben.

5.3.1 Entwicklungsumgebung

Quelle: Aufgabenstellung, Workshop

Der Fokus liegt klar auf Eclipse, da es sehr verbreitet ist und von der Schule in jedem Fach mit Bereichen in der Software Entwicklung verwendet wird. Beinahe jeder Student kennt und/oder hat schon einmal damit gearbeitet.

Eine alternative wäre noch NetBeans, inwiefern der Fokus auf diese Entwicklungsumgebung erweitert werden kann wird im Laufe des Projektes festgestellt.

5.3.2 Version Control System

Quelle: Workshop

Zur Analyse beigezogen wird SVN und git. SVN auf Grund seiner Verbreitung und Beliebtheit und git, da es in der Schule gelehrt wird und auch allgemein bekannt ist. Die beiden Tools werden gegeneinander analysiert und anschliessend wird eine Empfehlung abgegeben, der Student kann sich dann selber entscheiden.

5.3.3 Analyse der Code Metriken

Quelle: Projektteam

Die Tools die zur Analyse der Code Metriken dienen werden hier analysiert. Ziel ist es ein Add-On oder etwas Ähnliches für die Entwicklungsumgebung zu finden, welche alle wichtigen Metriken einfach und übersichtlich analysiert und je nach Möglichkeit auch exportierbar macht.

5.3.4 Projektplan

Quelle: Projektteam, Workshop

Die Planung des Projektes soll zu Beginn nur noch ganz grob gemacht werden mit der Planung der einzelnen Phasen und Meilensteine. Dies erfolgt in einem Dokument, welches unter Kapitel 2.1.2 beschrieben ist.

Bei diesem Punkt geht es darum Tools die bei der Unterstützung des Projektmanagements helfen und diese zu analysieren. Wichtige Punkte sind bei einem solchen Tool folgende:

- Meilensteine erfassbar
- Arbeitspakete werden Meilensteinen zugeteilt, es kann ein Soll-Aufwand für das Arbeitspaket erfasst werden und das Arbeitspaket kann einem Projektmitarbeiter zugeteilt werden.
- Der Projektmitarbeiter kann seine Arbeitspakete dokumentieren (Kommentare, Probleme) und sofern möglich direkt seine Zeit auf dem Paket erfassen.
- Für alles sollten gute Exports gemacht werden können (Meilenstein und Arbeitspaketübersicht und Zeiterfassung der einzelnen Projektmitarbeiter)

5.3.5 Zeiterfassung

Quelle: Projektteam

Die Zeiterfassung soll sofern möglich kombiniert werden mit dem Projektplan und den Arbeitspaketen, so dass der Student bei der Erfassung weniger Aufwand hat. Dies ist schon unter Kapitel 2.2.4 beschrieben.

5.3.6 Bug Tracking

Quelle: Workshop

Das Bug Tracking sollte wenn möglich im Tool unter Kapitel 2.2.4 integriert sein. Bugs werden dann erfasst wie Arbeitspakete und einem Projektmitarbeiter zugeteilt.

5.3.7 Design von User Interfaces

Quelle: Projektteam

Ein Tool welches das Design unterstützt (Mockups) soll dem Studenten nahegelegt werden. Dieses Tool soll das Design von User Interfaces für Bildschirme aber auch mobile Clients ermöglichen.

5.3.8 Buildserver

Quelle: Workshop

Die Analyse verschiedener Buildserver wird hier durchgeführt (Jenkins, TFS). Dem Student wird hier eine Auswahl dem Image des Servers hinzugefügt.

5.4 Beispielprojekt

Quelle: Aufgabenstellung, Workshop

Auf Grund der neu definierten Vorlagen wird ein Beispielprojekt in Java erstellt. Dieses sollte allen Anforderungen vom Modul Software Engineering 2 entsprechen. Das Projekt wird sogleich auf der neu empfohlenen und erstellten Projektumgebung integriert werden, damit die Studenten sich ein Bild machen können, was konkret verlangt wird. Dies vor allem auch bezüglich Code Design, Test Driven Development, Automatische Builds und Anordnung des Projektes.

6. Evaluation

6.1 Entwicklungsumgebung

6.1.1 Zweck

Dieses Dokument beschreibt die Beurteilung der beiden Entwicklungsumgebungen von Eclipse und NetBeans.

6.1.2 Quellen

Folgend sind die Quellen dieses Dokumentes aufgelistet:

- Anforderungsspezifikation.pdf
- Eclipse (http://www.eclipse.org/)
- NetBeans (http://netbeans.org/index.html)
- Verwendete Artikel:
  - http://www.retokiefer.com/archives/ide-shootout-2011-eclipse-vs-netbeans-vs-intellij-idea/ (Stand 30.09.12)
  - http://www.devx.com/Java/Article/34009/0/page/1 (Stand 30.09.12, Die einzelnen Seiten können unterhalb der Einführung durch gesehen werden.)

6.1.3 Tools

Es geht hier um Evaluation der Entwicklungsumgebungen Eclipse und NetBeans.

Bei der Evaluation wurde der vorgängig in der Anforderungsspezifikation definierte Funktionsumfang analysiert und ausgewertet.

6.1.3.1 Eclipse

6.1.3.1.1 Allgemein

Eclipse ist ein kostenfreies Programmierwerkzeug zur Entwicklung von Software verschiedenster Art. Die Entwicklung in verschiedensten Programmiersprachen ist damit möglich. Weitere Informationen finden Sie unter http://de.wikipedia.org/wiki/Eclipse_%28IDE%29 oder unter http://www.eclipse.org/.

Dieses Tool wird von der Hochschule für Technik Rapperswil in diversen Modulen verwendet und die Studenten kommen schon früh im Studium damit in Kontakt. Auch in der Branche ist dieses Tool weit verbreitet.

6.1.3.1.2 Analyse & Fazit

Dieses Tool ist sehr einfach zu verwenden, kostenfrei und leicht zu installieren. Ein grosser Vorteil von Eclipse sind seine vielen Plug-Ins. Ausserdem hat man rasch einen Überblick über die Funktionalitäten und kann diese auch sogleich anwenden. Gewisse Menüpunkte oder Funktionen sind im Menü etwas versteckt oder an unklaren Orten definiert (z.B. Updates sind unter dem Menüpunkt *Help* definiert).

Eclipse wird auch von der HSR in allen Modulen mit Thema oder Bereichen in der Software Entwicklung verwendet, weshalb sich die Studenten damit schon auskennen.

6.1.3.2 NetBeans

6.1.3.2.1 Allgemein

NetBeans ist eine Entwicklungsumgebung, die komplett auf der Programmiersprache Java entwickelt wurde und auf der NetBeans Plattform läuft. Diese Umgebung wurde hauptsächlich für die Programmiersprache Java entwickelt, unterstützt jedoch auch andere Programmiersprachen. Weitere Informationen finden Sie unter http://de.wikipedia.org/wiki/NetBeans_IDE oder unter http://netbeans.org/index.html.

NetBeans ist ebenfalls eine kostenfreie Anwendung und geniesst ein ständiges Wachstum an Nutzern.

6.1.3.2.2 Analyse & Fazit

NetBeans ist vor allem im Bereich der GUI Entwicklung (Swing) besser. Gute GUI Builder sind bei NetBeans gratis. Bei Eclipse ist der GUI Builder ebenfalls gratis, jedoch viel mühsamer in der Anwendung.

Die Verwendung dieses Tools ist immer mehr verbreitet. An der HSR wird jedoch meistens mit Eclipse gearbeitet. Studenten die jedoch damit schon gearbeitet haben oder es kennen lernen möchten steht nichts im Wege. NetBeans bietet ebenfalls eine sehr gute Entwicklungsumgebung und viele Plug-Ins und Tools.

Empfehlungen für Plug-Ins oder Tools für NetBeans werden keine abgegeben.

6.1.4 Gegenüberstellung & Auswahl

Auf eine Gegenüberstellung im direkten Sinn wurde in diesem Fall verzichtet, da Eclipse von der Schule in jedem Modul, wo es um Software Entwicklung geht, verwendet wird und es auch in der Branche ein sehr verbreitetes und beliebtes Werkzeug ist. Es macht keinen Sinn den Studenten beim Software Engineering Projekt eine andere Software zu empfehlen, als diese die von der Schule verwendet wird. Somit wird **Eclipse** empfohlen.

Natürlich können die Studenten auch NetBeans verwenden. Da gewisse dieses Tool schon gut kennen und ausschliesslich damit arbeiten. Den Studenten wird ebenfalls NetBeans vorgestellt und es werden auch Empfehlungen und Hinweise zu diesem Tool abgegeben.

NetBeans wird auch empfohlen, wenn eine Applikation massiv GUI-lastig ist, da NetBeans die besseren GUI Builder bietet. Die GUI Builder von Eclipse sind auch zu gebrauchen, jedoch mühsamer und umständlicher in der Anwendung.

6.2 Metrik Tools

6.2.1 Zweck

Dieses Dokument beschreibt die Evaluation und anschliessende Auswahl eines oder mehrerer Metrik Tools für Eclipse. Es soll aufzeigen aus welchen Gründen eine entsprechende Wahl getroffen wurde.

6.2.2 Quellen

Folgend sind die Quellen dieses Dokumentes aufgelistet:

- Anforderungsspezifikation.pdf
- Eclipse Marketplace (http://marketplace.eclipse.org/; zur Suche von Eclipse Metrik Tools)
- Vorlesungsunterlagen SE2 (um wichtige Bereiche der Metrikenanalyse aufzuzeigen und Vorschläge für zu verwendende Tools)
- Eclipse Metrics (http://www.stateofflow.com/projects/16/eclipsemetrics)
- Eclipse Metrics Plugin (http://metrics.sourceforge.net/)

6.2.3 Tools

Es geht hier um die Analyse von Metrik Tools, welche die Produktmetriken eines Software Engineering Projektes analysieren. Structure101, welches im Modul Software Engineering 2 behandelt wurde, wurde nicht analysiert, da man nur eine 30-tägige Lizenz erhält und danach wird das Tool kostenpflichtig.

Schlussendlich wurden zwei Plug-Ins von Eclipse analysiert, welche die besten Bewertungen im Eclipse Marketplace geniessen.

Bei der Evaluation wurde der vorgängig in der Anforderungsspezifikation definierte Funktionsumfang analysiert und ausgewertet.

6.2.3.1 Eclipse Metrics

6.2.3.1.1 Allgemein

Dies ist ein kostenfreies Plug-In für Eclipse und berechnet verschiedene Metriken. Es beinhaltet ebenfalls ein Warnsystem, welches in der Problem View von Eclipse auf Probleme hinweist. Somit kann man während der Entwicklung des Codes den Überblick behalten und die relevanten Metriken einhalten. Das Exportieren der Metriken in verschiedene Formate ist ebenfalls möglich.

6.2.3.1.2 Installation

Die Installation diese Plug-Ins gestaltet sich als sehr einfach. Folgende Punkte wurden dazu durchgeführt:

1. Starten von Eclipse
2. Über *Help > Eclipse Marketplace..* die Suche nach Eclipse Metrics starten
3. Danach auf *Install* klicken und das Plug-In wird automatisch installiert

Mehr muss dazu nicht gemacht werden und das Plug-In ist im Eclipse installiert und kann sofort verwendet werden. Die Installation dauert je nach Internetverbindung etwas länger, da die Installationspakete von einem externen Server heruntergeladen werden müssen.

6.2.3.1.3 Benutzbarkeit

Um die Metrikenanalyse für ein laufendes Projekt einzuschalten muss dies in den Projekt Eigenschaften unter *Metrics* aktiviert werden.

Unter den allgemeinen Eigenschaften unter dem Punkt *Metrics* können zusätzlich die einzelnen Grenzwerte definiert werden. Was es dem Benutzer einfach macht eigene Grenzwerte zu definieren, welche zum Beispiel durch einen Dozenten oder einen Projektleiter vorgegeben werden.

Tritt irgendwo ein Problem auf, so wird dies in der *Problem* View von Eclipse und man sieht es auch direkt im Code durch eine rote Markierung am Rande. Die folgenden zwei Abbildungen zeigen dies.

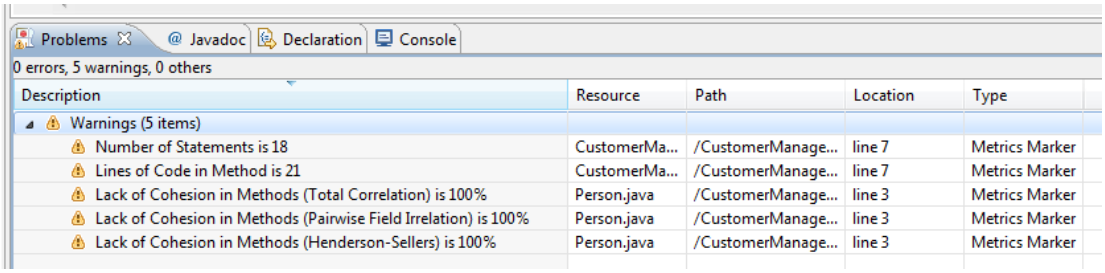


Abbildung 9: Evaluation Metrik Tools: EclipseMetrics: Problem View im Eclipse

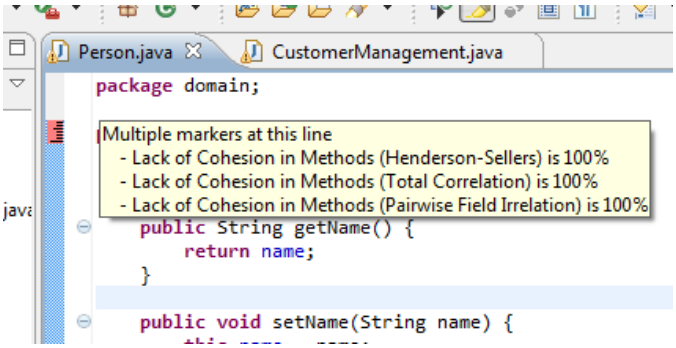


Abbildung 10: Evaluation Metrik Tools: EclipseMetrics :Direkt im Code

Die Benutzung dieses Plug-Ins ist wie man sieht sehr einfach. Eine einfache Installation und ein Aktivieren genügt um die Metriken seines Codes analysieren zu können.

6.2.3.1.4 Exportierbarkeit

Die einzelnen errechneten Werte können in verschiedene Formate exportiert werden (XML, CSV und HTML). Die HTML Version ist sehr übersichtlich und man hat direkte Auswertungen der einzelnen Metriken in grafischer Form mit den dazugehörigen Referenzen im Code. Die CSV Version listet die jeweiligen Messwerte der Metriken auf, müsste aber noch weiterverarbeitet werden um brauchbare Grafiken und Übersichten zu bekommen.

6.2.3.1.5 Fazit

Dieses Tool ist sehr einfach zu verwenden, kostenfrei und leicht zu installieren. Man hat rasch einen Überblick über die Funktionalitäten und kann diese auch sogleich anwenden. Durch die direkte Unterstützung in Eclipse mit der Fehleranzeige wird der Programmierer schon beim Codieren auf Metriken hingewiesen und der Lerneffekt ist somit viel grösser als wenn dies erst am Ende der Codierung passiert.

6.2.3.2 Eclipse Metrics Plugin

6.2.3.2.1 Allgemein

Eclipse Metrics Plugin ist ein weiteres Plug-In zur Analyse der Metriken für Eclipse. Es bietet Ähnlichen Komfort wie Eclipse Metrics ist jedoch anders zu konfigurieren.

6.2.3.2.2 Installation

Die Installation dieses Plug-Ins gestaltet sich ebenfalls als sehr einfach. Folgende Punkte wurden dazu durchgeführt:

1. Starten von Eclipse
2. Über *Help > Install New Software..*
3. Den entsprechenden Link eingeben und das Plug-In auswählen.
4. Danach die Installation durchführen.

Mehr muss dazu nicht gemacht werden und das Plug-In ist im Eclipse installiert und kann sofort verwendet werden.

6.2.3.2.3 Benutzbarkeit

Um die Metrikenanalyse für ein laufendes Projekt einzuschalten muss dies ebenfalls in den Projekt Eigenschaften unter *Metrics* aktiviert werden.

Unter den allgemeinen Eigenschaften unter dem Punkt *Metrics Preferences* müssen noch die einzelnen Grenzwerte definiert werden.

Tritt irgendwo ein Problem auf, so wird dies in der *Metrics* View von Eclipse. Man sieht jedoch im Code direkt keine speziellen Markierungen oder Hinweise. Die folgende Abbildung zeigt diese *Metrics* View.

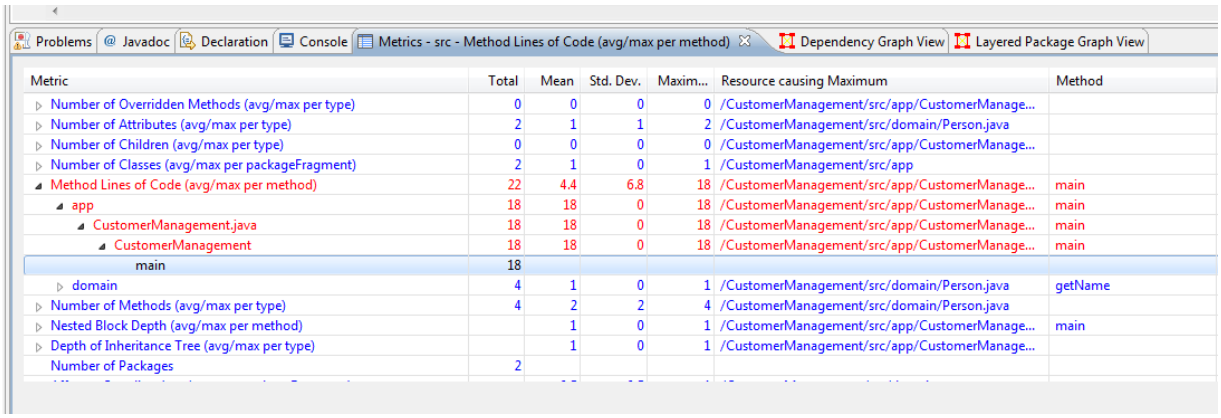


Abbildung 11: Evaluation Metrik Tools: EclipseMetrics Plugin: Metrics View im Eclipse

Des Weiteren bietet dieses Plug-In noch eine kleine Übersicht der Abhängigkeiten, welche aus der *Metrics* View generiert werden kann und dann in der *Dependency Graph View* angesehen werden kann. Folgende Abbildung zeigt einen solchen Abhängigkeitsbaum.

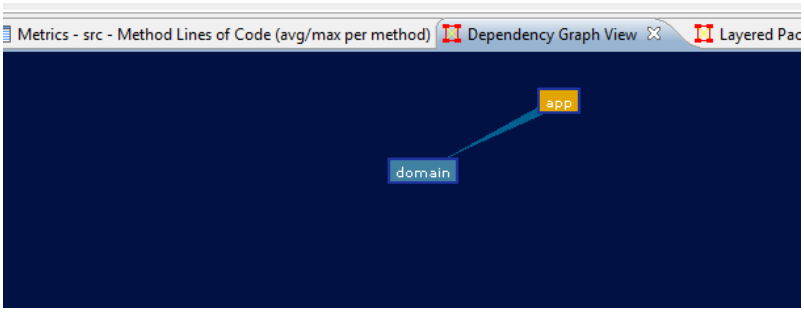


Abbildung 12: Evaluation Metrik Tools: EclipseMetrics Plugin: Abhängigkeitsbaum der Packages

Dieser Baum ist jedoch nicht sehr aussagekräftig und zeigt nur durch eine Linie wo Abhängigkeiten sind. Wie viele dies sind oder in welche Richtung diese Abhängigkeiten sind, ist in dieser Ansicht nicht ersichtlich.

6.2.3.2.4 Exportierbarkeit

Alle Metriken sind nach XML exportierbar, was zur Weiterverarbeitung sinnvoll ist. Eine Auswertung fällt so jedoch sehr schwer, da keine übersichtlichen Grafiken generiert werden und auch kein CSV Export oder ähnliches möglich ist.

6.2.3.2.5 Fazit

In der Benutzbarkeit und Exportierbarkeit wird es von Eclipse Metrics klar geschlagen, da das andere Tool mehr Funktionalitäten bietet. Einzig die Übersicht der Abhängigkeiten ist ein Vorteil dieses Tools. Da diese Übersicht aber wie beschrieben nicht sehr aussagekräftig ist und nicht sehr viele Möglichkeiten bietet, kann dieser Punkt vernachlässigt werden.

Zur Unterstützung des Programmierers während der Code Entwicklung dient es aber gut. Falls nur dies gefordert wäre, so könnte man auch dieses Plug-In verwenden, je nach dem was einem besser passt.

6.2.4 Gegenüberstellung

6.2.4.1 Bewertungspunkte

In den folgenden Abschnitten werden die einzelnen Bewertungspunkte beschrieben. Im abschliessenden Abschnitt erfolgt die Bewertung der einzelnen Tools bezüglich der Bewertungspunkte. Jeder Bewertungspunkt beinhaltet noch eine Gewichtung von 1 bis 10 je nach Wichtigkeit.

6.2.4.1.1 Wichtige Metriken

Dieser Punkt ist sehr wichtig. Er sagt nämlich aus ob die wichtigsten und sinnvollsten Metriken in den Tools überprüft werden. Folgende Metriken zählen zu den wichtigsten:

- CC (Cyclomatic Complexity): Anzahl Wege durch eine Methode (if, else, switch, …).
- WMC (Weighted Methods per Class): Komplexität von Methoden innerhalb einer Klasse.
- FE (Feature Envy): Methode interessiert sich mehr für Eigenschaften einer anderen Klasse.
- LCOM (Lack of Cohesion of Methods): Kohäsion und Kopplung von Methoden.
- LoC (Lines of Code): Anzahl Codezeilen einer Methode.
- NoP (Number of Parameters): Anzahl Parameter einer Methode.
- EC (Effrent Coupling): Anzahl Packages ausserhalb von denen Klassen im Package abhängen.
- NoS (Number of Statements): Anzahl Befehle innerhalb einer Methode.

6.2.4.1.2 Weitere Metriken

Zu den wichtigen gibt es auch noch eine grosse Anzahl weiterer Metriken, welche analysiert werden können. Diese scheinen aber nicht von gleich grossem Wert wie die bis anhin aufgezählten. Die nachfolgende Liste soll einen Einblick gewähren, welche Metriken es noch gibt:

- AC (Afferent Coupling): Packages ausserhalb, die von Klassen innerhalb abhängen.
- NoF (Number of Fields): Anzahl Felder einer Klasse.
- NoLS (Number of Locals in Scope): Anzahl lokaler Variablen innerhalb eines Bereiches.
- usw.

6.2.4.1.3 Einbindung

Eine Einbindung in die Entwicklungsumgebung in Form eines Plug-Ins des Tools ist ein Punkt der analysiert wird. Es vereinfacht einiges in der Auswertung, wenn man nicht noch zusätzliche Software installieren muss, sondern nur ein Plug-In für die Entwicklungsumgebung. Ausserdem muss man den kompilierten Source Code nicht nochmals mit einer anderen Software starten.

6.2.4.1.4 Änderbarkeit

Die einzelnen Grenzwerte der Metriken sollten anpassbar sein, damit je nach Vorgaben im Projekt mit den nötigen Grenzwerten gearbeitet werden kann.

6.2.4.1.5 Benutzung

Das Tool soll den Benutzer bei der Code Erstellung unterstützen und direkt Fehler im Zusammenhang mit den Metriken anzeigen, damit Probleme vom vorhinein vermieden werden können.

6.2.4.1.6 Auswertung

Eine saubere und übersichtliche Auswertung der Metriken ist wichtig. Man sollte direkt Grafiken, Excel-Files oder ähnliches exportieren können um die Daten auswerten zu können.

6.2.4.1.7 Abhängigkeiten

Eine Übersicht mit den Package Abhängigkeiten ist ein Punkt. In diesem Fall reicht eine kleine Grafik mit den einzelnen Packages und deren Abhängigkeiten untereinander.

6.2.4.2 Übersicht

		Eclipse Metrics			Eclipse Metrics Plugin		
	Gew.	**Wert.**	**Beschreibung**	**Total**	**Wert.**	**Beschreibung**	**Total**
2.1.1 Wichtige Metriken	10	10	Alle wichtigen Metriken sind abgedeckt.	100	10	Alle wichtigen Metriken sind abgedeckt.	100
2.1.2 Weitere Metriken	6	7	Weitere Metriken sind vorhanden, aber nicht viele (4-5).	42	10	Viele weitere Metriken sind vorhanden und auswertbar.	60
2.1.3 Einbindung	7	9	Einbindung möglich, auch relativ einfach.	63	8	Installation und Setup dauert etwas länger als beim anderen Plug-In.	56
2.1.4 Änderbarkeit	8	9	Alle Werte sind einstellbar.	72	10	Alle Werte sind einstellbar, sogar das Minimum kann gesetzt werden.	80
2.1.5 Benutzung	7	10	Alle Probleme werden in einer View angezeigt und auch direkt im Code.	70	8	Alle Probleme werden in einer View angezeigt, jedoch nicht direkt im Code.	56
2.1.6 Auswertung	9	8	HTML Auswertung gut und mit Grafiken. Die CSV Auswertung ist jedoch nicht so überzeugend.	72	2	Es ist lediglich eine XML Auswertung möglich. Kein CSV oder HTML und auch keine Grafiken können generiert werden.	18
2.1.7 Abhängigkeiten	8	0	Keine Auswertung mit Abhängigkeiten über die Packages möglich.	0	2	Es gibt eine Auswertung, diese ist jedoch nicht sehr aussagekräftig.	16
TOTAL				**418**			**386**

Tabelle 6: Evaluation Metrik Tools: Gegenüberstellung

6.2.4.3 Auswahl

Eclipse Metrics schlägt Eclipse Metrics Plugin nicht nur in der Gegenüberstellung sondern auch in der persönlichen Anwendung. Das Fazit von Eclipse Metrics ist deutlich besser ausgefallen als dies vom Eclipse Metrics Plugin.

Es wird deshalb empfohlen mit **Eclipse Metrics** zu arbeiten.

6.3 Strukturanalyse Tools

6.3.1 Zweck

Dieses Dokument beschreibt die Evaluation und anschliessende Auswahl eines Strukturanalyse Tools für Java. Es soll aufzeigen aus welchen Gründen eine entsprechende Wahl getroffen wurde. Es geht hierbei nur um die Strukturanalyse des Codes, also das Aufzeigen von Abhängigkeiten der einzelnen Pakete. Weitere Metriken Tools werden in einem separaten Dokument evaluiert.

6.3.2 Quellen

Folgend sind die Quellen dieses Dokumentes aufgelistet:

- Anforderungsspezifikation.pdf
- Vorlesungsunterlagen Modul Software Engineering 2
- STAN (http://stan4j.com/)
- Structure101 (http://www.headwaysoftware.com/#intro)
- JDepend (http://clarkware.com/software/JDepend.html)

6.3.3 Tools

6.3.3.1 STAN

6.3.3.1.1 Allgemein

STAN steht für Structure Analysis for Java (Struktur Analyse für Java) und ist eines der führenden Strukturanalyse Tools für Eclipse. STAN kann als Plug-In für Eclipse installiert werden, dies erlaubt es während der Codierung jederzeit Auswertungen durchzuführen und die bisherig getane Arbeit zu analysieren.

STAN ist bei einer Anwendung bis 500 Klassen kostenlos, was als ausreichend für ein Software Engineering 2 Projekt angesehen werden kann.

6.3.3.1.2 Installation

Die Installation ist relativ einfach und läuft folgendermassen:

5. Starten von Eclipse
6. Über *Help > Install New Software..*
7. Den entsprechenden Link (http://update.stan4j.com/ide) eingeben und auswählen der einzelnen Pakete.
8. Danach die Installation durchführen.

Mehr muss dazu nicht gemacht werden und das Plug-In ist im Eclipse installiert und kann sofort verwendet werden.

6.3.3.1.3 Benutzbarkeit

Um eine Analyse auszuführen muss auf dem Projekt *Run As* und dann *Structure Analysis* auswählen. In der STAN View, welche definiert ist und angezeigt werden kann sind dann die Auswertungen zu sehen.

Im Folgenden sehen Sie einen Abhängigkeitsbaum generiert mit STAN.

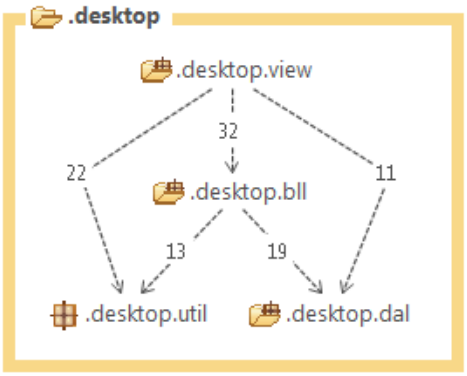


Abbildung 13: Evaluation Strukturanalyse Tools: STAN: Abhängigkeitsbaum

Hier kann auf das Detail der einzelnen Pakete eingegangen werden um mögliche Kreisabhängigkeiten zu finden.

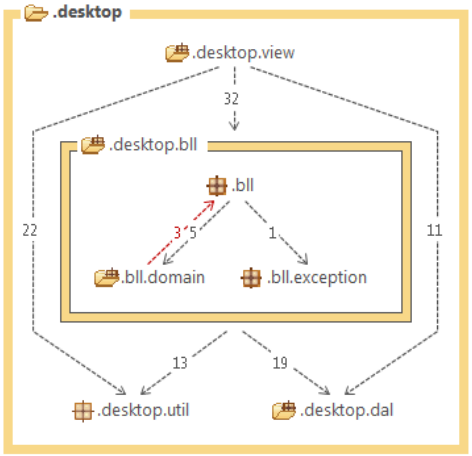


Abbildung 14: Evaluation Strukturanalyse Tools: STAN: Kreisabhängigkeiten

Die Möglichkeiten den Baum zu analysieren sind mit STAN viel umfangreicher, als mit Structure101. Als Beispiel, die Fähigkeit Details von einzelnen Paketen im Zusammenhang mit dem Ganzen darzustellen. Sichtbar in folgender Grafik:

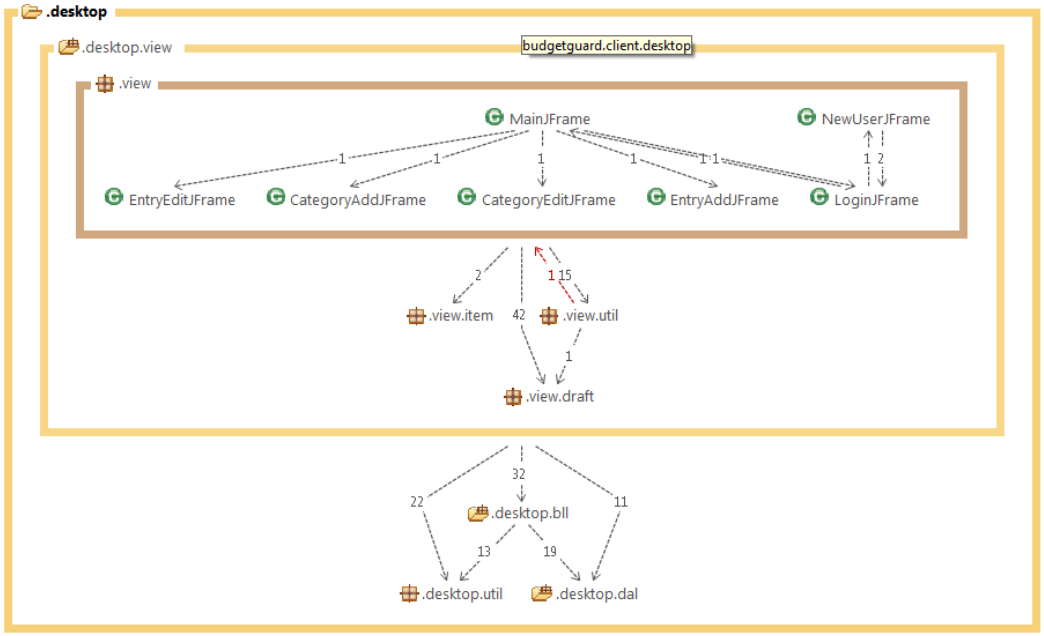


Abbildung 15: Evaluation Strukturanalyse Tools: STAN: Abhängigkeitsbaum detailliert

Somit kann man sehr viele Informationen in einem übersichtlichen Baum darstellen und auf mögliche Probleme direkt im Zusammenhang hinweisen.

6.3.3.1.4 Exportierbarkeit

Das Exportieren ist auch möglich. Es können Reports mit der *Export* Funktion von Eclipse generiert werden. Diese Reports beinhalten jedoch nur Metrikenanalyse und Kreisabhängigkeiten. Die einzelnen Pakete Abhängigkeiten sieht man nur direkt in Eclipse.

Will man diese Grafiken weiterverwenden, so muss ein Print Screen oder Snipping gemacht werden. Ein direktes Kopieren von der Grafik alleine ist leider nicht möglich.

6.3.3.1.5 Fazit

Das Tool ist bis zu einer Menge von 500 Klassen kostenlos und liegt somit in einem akzeptablen Bereich für ein Software Engineering 2 Projekt. Die Grafiken bieten ausserdem viele schöne Darstellungsmöglichkeiten und durch die direkte Integration in Eclipse können diese Auswertungen auch schon von Anfang an verwendet werden.

6.3.3.2 Structure101

6.3.3.2.1 Allgemein

Structure101 ist ein kommerzielles Tool zur Struktur- und Komplexitätsanalyse. Geeignet ist es für die Programmiersprachen Java und .Net Programmiersprachen. Das Tool berechnet Abhängigkeiten im Code auf verschiedenen Strukturelementen.

Für Structure101 gibt es jedoch nur eine kostenlose Lizenz für 30 Tage, danach wird das Produkt kostenpflichtig.

6.3.3.2.2 Installation

Die Installation gestaltet sich als sehr einfach und kann nach folgenden Punkten durchgeführt werden:

1. Die Software lässt sich über die Homepage downloaden unter der Auswahl der gewünschten Programmiersprache und des Systems auf welchem die Software laufen soll.
2. Danach die Installation mit Hilfe der .exe Datei durchführen.
3. Beim ersten Start muss eine Lizenzdatei (.lic) angegeben werden, welche man per Mail erhält (Diese Datei enthält die 30 Tage gültige Lizenz).

Danach kann das Tool während 30 Tagen kostenfrei verwendet werden.

6.3.3.2.3 Benutzbarkeit

Um eine Strukturanalyse durchzuführen kann man den Pfad zu den Klassen angeben oder aber auch direkt eine Eclipse Workspace auswählen. Nach der Auswahl des Projektes kann man noch weitere Einstellungen vornehmen (z.B. wie detailliert die Auswertung sein soll, usw.).

Im Folgenden sehen Sie einen Abhängigkeitsbaum, der von Structure101 generiert wurde. Er zeigt auf der obersten Paketebene alle Abhängigkeiten.

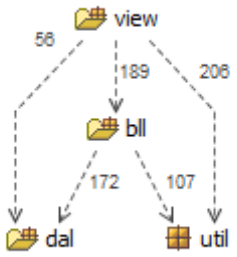


Abbildung 16: Evaluation Strukturanalyse Tools: Structure101: Abhängigkeitsbaum

Man kann auch die einzelnen Pakete analysieren und sieht so auch direkt unschöne Kreisabhängigkeiten. Folgendes Beispiel ist die Detailansicht vom bll Paket, welches Kreisabhängigkeiten enthält.

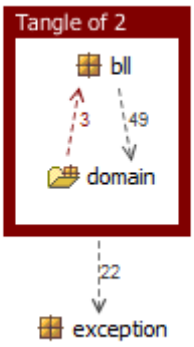


Abbildung 17: Evaluation Strukturanalyse Tools: Structure101: Kreisabhängigkeiten

Auch Zusammenhänge können aufgezeigt werden, was folgendes Beispiel ersichtlich macht.

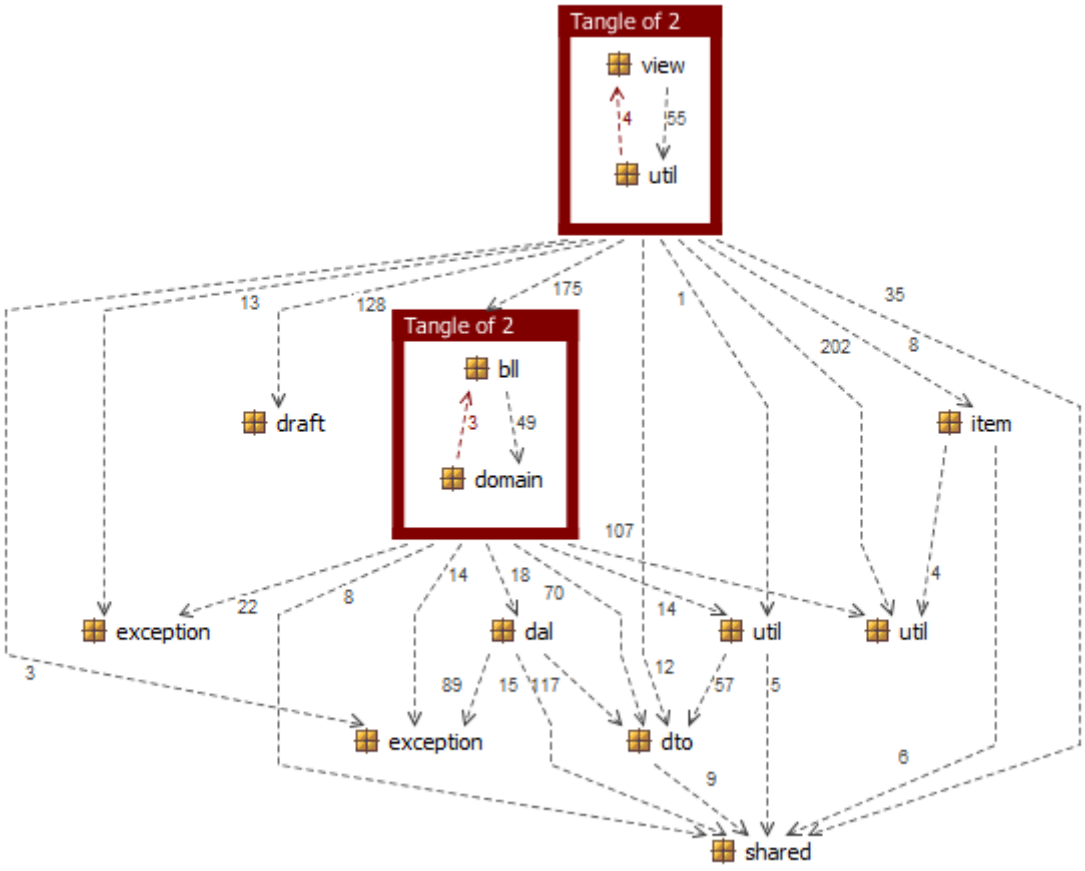


Abbildung 18: Evaluation Strukturanalyse Tools: Structure101: Abhängigkeitsbaum detailliert

6.3.3.2.4 Exportierbarkeit

Die einzelnen Elemente sind einfach zu kopieren aus der Software und können problemlos in anderen Programmen eingebunden werden, wie z.B. Word. Reports im Format PDF oder ähnliches können jedoch nicht generiert werden. Man kann jedoch leicht die relevanten Bereiche finden (wie z.B. in den obigen Bildern dargestellt) und diese in ein Dokument überführen und genauer beschreiben.

6.3.3.2.5 Fazit

Das Tool ist leicht anzuwenden, bietet allerdings mehr Funktionalitäten als nötig. Ein Problem ist die nur 30 Tage dauernde kostenfreie Lizenz. Die Studenten könnten somit das Tool erst in den letzten 30 Tagen der Software Entwicklung und im Projekt verwenden um ihre Struktur zu analysieren und nicht von Beginn an. Ansonsten bietet das Tool alles was nötig ist.

6.3.3.3 JDepend

JDepend war auch noch ein Vorschlag. Dieses Tool wäre gratis, da die Anwendung aber nur über die Konsole möglich ist und keine schönen Grafiken über die Abhängigkeiten generierbar sind, wurde dieses Tool aus der Auswahl entfernt.

Für eine Konsolenanwendung könnte die Einbindung in einem Buildserver (z.B. Jenkins) sprechen. Es wird noch überprüft, ob dies Sinn macht und möglich ist.

6.3.4 Gegenüberstellung

6.3.4.1 Bewertungspunkte

In den folgenden Abschnitten werden die einzelnen Bewertungspunkte beschrieben. Im abschliessenden Abschnitt erfolgt die Bewertung der einzelnen Tools bezüglich der Bewertungspunkte. Jeder Bewertungspunkt beinhaltet noch eine Gewichtung von 1 bis 10 je nach Wichtigkeit.

6.3.4.1.1 Abhängigkeiten

Es muss möglich sein die Abhängigkeiten der Pakete mit Detailinformationen anzusehen. Das heisst man kann auf ein Problem klicken und sieht direkt im Code wo dieser Aufruf entsteht.

6.3.4.1.2 Details

Die Darstellung der Abhängigkeitsbäume sollte auch detaillierter möglich sein um zum Beispiel den Zusammenhang der Pakete mit Detailview auf ein Paket zu ermöglichen. Dies dient dazu um spezielle Probleme besser erklären zu können.

6.3.4.1.3 Kosten

Das Tool sollte kostenfrei sein.

6.3.4.1.4 Benutzung

Eine Einbindung in Eclipse wäre von Vorteil um die Verwendung des Tools zu vereinfachen. Es sollte ausserdem leicht benutzbar sein und den Studenten keine Mühe bereiten.

6.3.4.1.5 Exportierbarkeit

Die Grafiken sollten leicht exportierbar sein um sie in anderen Tools weiterverwenden zu können (wie z.B. in Word).

6.3.4.2 Übersicht

		STAN			Structure101		
	Gew.	**Wert.**	**Beschreibung**	**Total**	**Wert.**	**Beschreibung**	**Total**
2.1.1 Abhängig-keiten	10	10	Abhängigkeitsbaum verfügbar.	100	10	Abhängigkeitsbaum verfügbar.	100
2.1.2 Details	6	10	Details im Zusammenhang ersichtlich und viele Anzeigemöglichkeiten.	60	8	Details auch im Zusammenhang ersichtlich, bietet jedoch weniger Möglichkeiten als STAN.	48
2.1.3 Kosten	10	6	Bis 500 Klassen kostenlos, ist vertretbar.	60	4	30 Tage kostenlos. Wäre nur am Ende brauchbar.	40
2.1.4 Benutz-ung	8	9	Einbindung in Eclipse möglich und leicht anzuwenden.	72	6	Einbindung möglich aber nur im Zusammenhang mit gemachten Auswertungen im Standalone Produkt.	48
2.1.5 Exportier-barkeit	7	7	Grafiken nur mit Print Screens exportierbar.	49	10	Grafiken einfach zu kopieren und in anderer Software weiterverwendbar.	70
TOTAL				**341**			**306**

Tabelle 7: Evaluation Strukturanalyse Tools: Gegenüberstellung

6.3.4.3 Auswahl

STAN schlägt Structure101 nicht nur in der Gegenüberstellung sondern auch in der persönlichen Anwendung. Durch die leichte Einbindung in Eclipse und die Möglichkeit das Tool während der gesamten Entwicklung zu verwenden ist STAN besser geeignet. Structure101 wäre nur für die letzten 30 Tage der Software Entwicklung einsetzbar. Dass bei STAN ein Projekt nur 500 Klassen beinhalten darf um kostenfrei Auswertungen machen zu können wird für ein Software Engineering 2 Projekt als realistisch angesehen.

Es wird deshalb empfohlen mit **STAN** zu arbeiten.

6.4 User Interface Tools

6.4.1 Zweck

Dieses Dokument soll einen GUI-Builder für Eclipse empfehlen. Da Eclipse als Entwicklungsumgebung ausgewählt wurde, soll den Studenten einen guten brauchbaren GUI-Builder empfohlen werden.

6.4.2 Quellen

Folgend sind die Quellen dieses Dokumentes aufgelistet:

- Anforderungsspezifikation.pdf
- EvaluationEntwicklungsumgebung.pdf
- Artikel mit Vergleich von GUI Builder für Eclipse Stand: 18.10.2011 (http://wiki.computerwoche.de/doku.php/programmierung/gui-builder_fuer_eclipse)
- WindowBuilder Pro (http://code.google.com/intl/de-DE/javadevtools/wbpro/index.html)

6.4.3 Tools

Aufgrund der riesigen Auswahl an UI-Tools, mehrere Tools für jede Art der Entwicklung(Swing, AWT, JGoodies, HTML, usw.), wird auf eine spezifische Empfehlung verzichtet.

Es wird lediglich ein geeigneter Swing GUI Builder wird für Eclipse empfohlen. Jedoch wird auch hier auf einen Vergleich von verschiedenen GUI-Builder verzichtet, da das Angebot viel zu gross ist. Es werden die Vergleiche der Quelle analysiert.

Nach der Analyse der Vergleiche von obig genannter Quelle ist der Entscheid auf den WindowBuilder Pro gefallen, der von Google gratis angeboten wird.

6.4.3.1 WindowBuilder Pro

6.4.3.1.1 Allgemein

WindowBuilder Pro ist ein mächtiges und einfach benutzbares Java GUI Design Tool. Weitere Informationen dazu finden Sie unter den Links, welche als Quellen angegeben wurden.

Ein neues JFrame, JPanel, usw. kann direkt im Eclipse New Wizard erstellt werden, wie die folgende Abbildung zeigt.

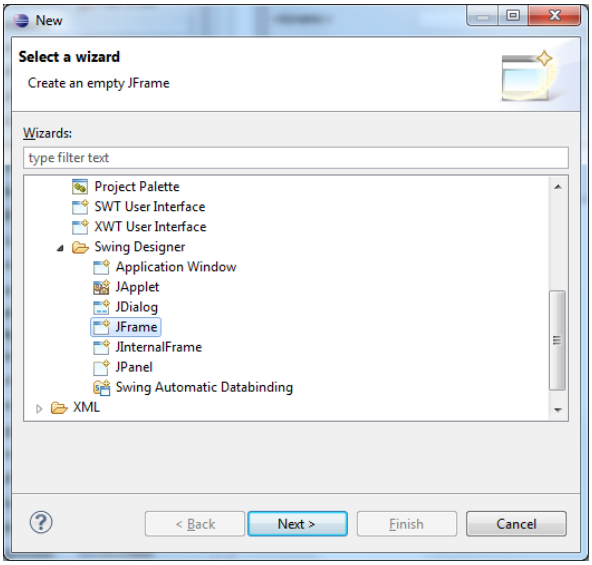


Abbildung 19: Evaluation User Interface Tools: Neuerstellung

Folgender Screenshot zeigt wie einfach es ist einzelne Elemente mit diesem GUI Builder zu generieren und im Fenster zu erstellen. Auch wenn gewisse Daten dynamisch herbeigeholt werden.

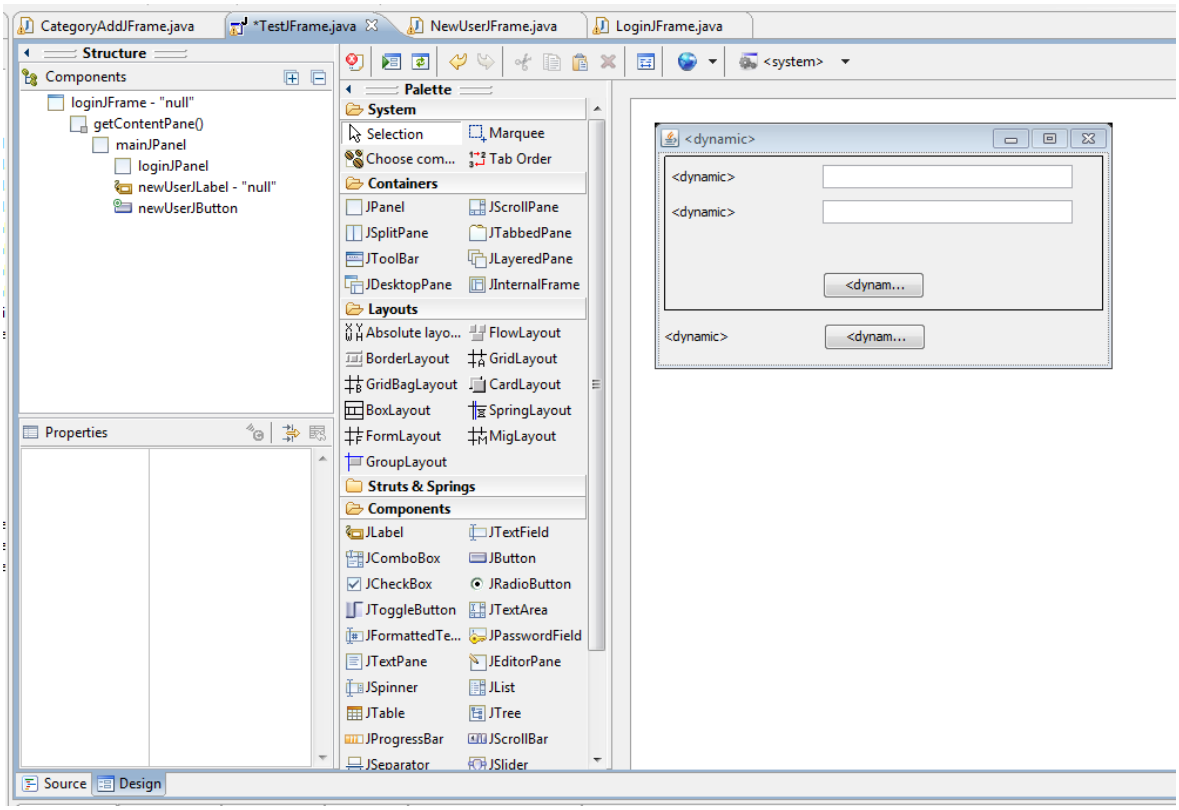


Abbildung 20: Evaluation User Interface Tools: Design View

6.4.3.1.2 Fazit & Empfehlung

Dieser Builder ist leicht zu installieren und bietet dem Benutzer viele Möglichkeiten. Auch die Anwendung ist einfach und gut dokumentiert. Auch die Bewertungen dazu sind gut (siehe unter Quellen den Artikel mit den Vergleichen).

Deshalb wird dieser GUI Builder empfohlen.

6.5 Weitere Plug-Ins für Eclipse

6.5.1 Zweck

Dieses Dokument beschreibt die Evaluation einer Auswahl verschiedener weiterer Plug-Ins für Eclipse und gibt, falls nötig, eine direkte Empfehlung ab. Auf eine Gegenüberstellung wird verzichtet, die einzelnen Tools werden aufgelistet und bei Bedarf empfohlen.

6.5.2 Quellen

Folgend sind die Quellen dieses Dokumentes aufgelistet:

- Anforderungsspezifikation.pdf
- Vorlesungsunterlagen Modul Software Engineering 2 (um weitere Tools zu finden)
- Eclipse Marketplace (http://marketplace.eclipse.org/; zur Suche von sinnvollen Tools)
- FindBugs (http://findbugs.sourceforge.net/)
- EclEmma (http://www.eclemma.org/)
- Checkstyle (http://checkstyle.sourceforge.net/index.html)

6.5.3 Tools

6.5.3.1 FindBugs

6.5.3.1.1 Allgemein

FindBugs ist ein Code Analyse Tool der University of Maryland. Es handelt sich um eine statische Analyse von Java Byte Code. FindBugs sucht in Java Programmen nach Fehlermustern (z.B. sind Überprüfungen korrekt, Variablen richtig initialisiert, usw.). Es handelt sich um ein kostenfreies Tool. Solche Fehlermuster deuten meist auf tatsächliche Fehler hin. Weitere Informationen dazu finden Sie unter http://de.wikipedia.org/wiki/FindBugs oder unter http://findbugs.sourceforge.net/.

6.5.3.1.2 Installation

Die Installation ist relativ einfach und läuft folgendermassen:

9. Starten von Eclipse
10. Über *Help > Install New Software…*
11. Den entsprechenden Link (http://findbugs.cs.umd.edu/eclipse) eingeben und Paket auswählen.
12. Danach die Installation durchführen.

Mehr muss dazu nicht gemacht werden und das Plug-In ist in Eclipse installiert und kann sofort verwendet werden.

6.5.3.1.3 Benutzbarkeit

Um FindBugs in Eclipse verwenden zu können muss im Kontextmenü *FindBugs > Find Bugs* ausgeführt werden und in der *Bug Explorer* View werden die entsprechenden Fehler angezeigt.

Im Folgenden sehen Sie einen Ausschnitt aus der Bug Explorer View von FindBugs.

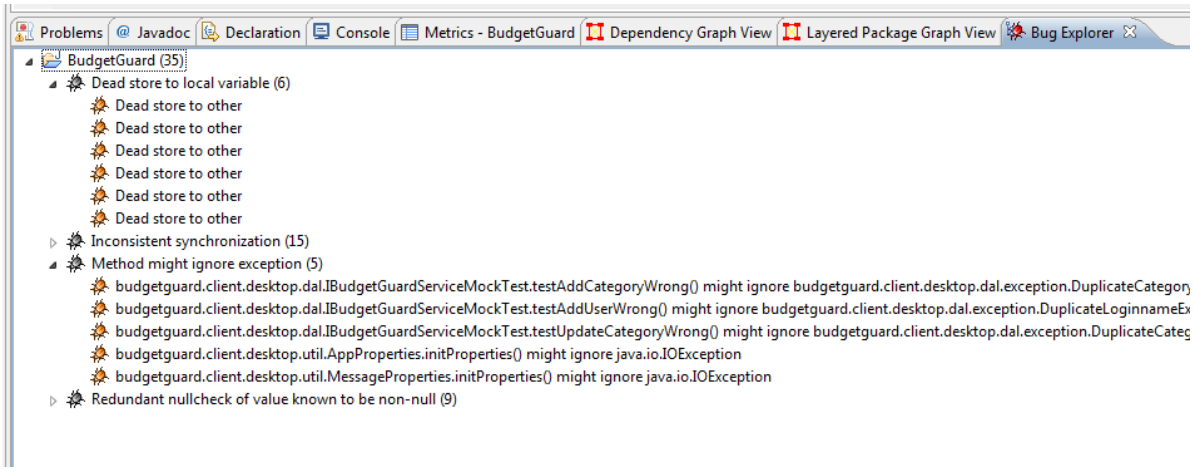


Abbildung 21: FindBugs: Bug Explorer

Auf die einzelnen Hinweise kann direkt geklickt werden und man kommt an die entsprechende Stelle im Code.

Die Fehler werden auch direkt im Code angezeigt, wenn dies über die Projekteigenschaften unter FindBugs eingestellt wurde. Das folgende Bild zeigt diese Anzeige.

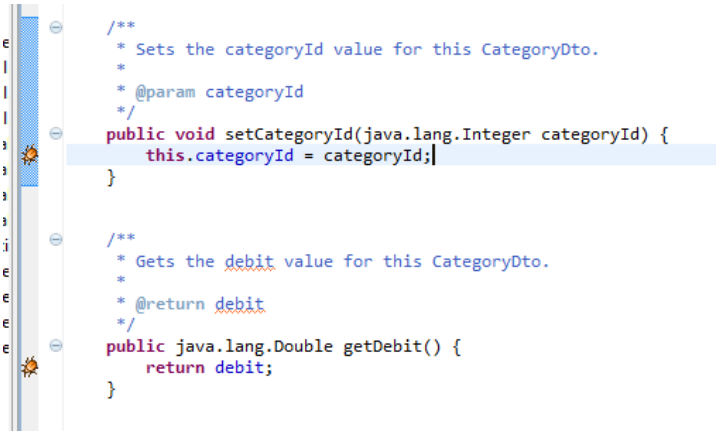


Abbildung 22: FindBugs: Direkt im Code

In den Projekteigenschaften können noch diverse Einschränkungen vorgenommen und Einstellungen angepasst werden.

6.5.3.1.4 Fazit & Empfehlung

FindBugs weist den Nutzer auf potentielle Fehler im Code hin, bevor der Code zur Ausführung kommt. Dieses Tool bietet dadurch eine gute Unterstützung während der Software Entwicklung. Es ist kostenfrei und sehr leicht zu bedienen.

Deshalb wird es den Studenten empfohlen.

6.5.3.2 EclEmma

6.5.3.2.1 Allgemein

EclEmma ist ein kostenfreies Tool zur Messung der Testabdeckung in Java Programmen und kann als Plug-In von Eclipse verwendet werden. Dadurch wird bei der Ausführung von Tests gemessen, durch welche Klassen, Methoden, Blöcke und Zeilen Code die Abarbeitung durchläuft. Weitere Informationen finden Sie unter http://www.eclemma.org/.

6.5.3.2.2 Installation

Die Installation diese Plug-Ins gestaltet sich als sehr einfach. Folgende Punkte wurden dazu durchgeführt:

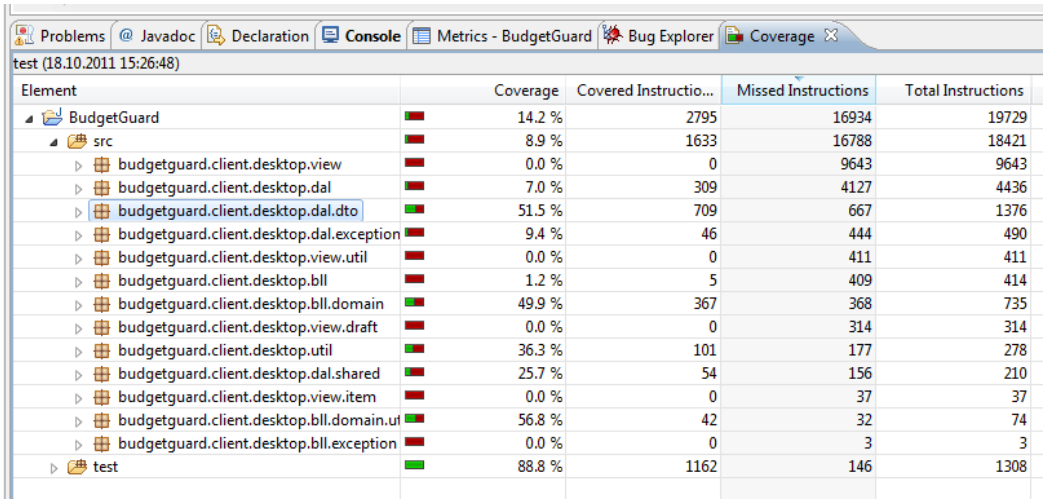
4. Starten von Eclipse
5. Über *Help > Eclipse Marketplace..* die Suche nach EclEmma starten
6. Danach auf *Install* klicken und das Plug-In wird automatisch installiert

Mehr muss dazu nicht gemacht werden und das Plug-In ist in Eclipse installiert und kann sofort verwendet werden.

6.5.3.2.3 Benutzbarkeit

Um die Testabdeckung zu testen müssen die Tests zuerst alle ausgeführt werden. Anschliessend kann deren Coverage berechnet werden. Genauere Anleitungen zur Benutzung finden Sie hier http://www.eclemma.org/userdoc/index.html.

Die folgende Abbildung zeigt ein mögliches Resultat in der Coverage View von EclEmma. Man sieht sofort, welche Klassen oder Pakete gut abgedeckt sind von den Tests und welche nicht.



Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
BudgetGuard	14.2 %	2795	16934	19729
src	8.9 %	1633	16788	18421
budgetguard.client.desktop.view	0.0 %	0	9643	9643
budgetguard.client.desktop.dal	7.0 %	309	4127	4436
budgetguard.client.desktop.dal.dto	51.5 %	709	667	1376
budgetguard.client.desktop.dal.exception	9.4 %	46	444	490
budgetguard.client.desktop.view.util	0.0 %	0	411	411
budgetguard.client.desktop.bll	1.2 %	5	409	414
budgetguard.client.desktop.bll.domain	49.9 %	367	368	735
budgetguard.client.desktop.view.draft	0.0 %	0	314	314
budgetguard.client.desktop.util	36.3 %	101	177	278
budgetguard.client.desktop.dal.shared	25.7 %	54	156	210
budgetguard.client.desktop.view.item	0.0 %	0	37	37
budgetguard.client.desktop.bll.domain.ut	56.8 %	42	32	74
budgetguard.client.desktop.bll.exception	0.0 %	0	3	3
test	88.8 %	1162	146	1308

Abbildung 23: EclEmma: Coverage View

Mit einem Doppelklick auf eine Klasse wird diese geöffnet und es ist sofort ersichtlich, welche Bereiche durch Tests abgedeckt sind und welche nicht.

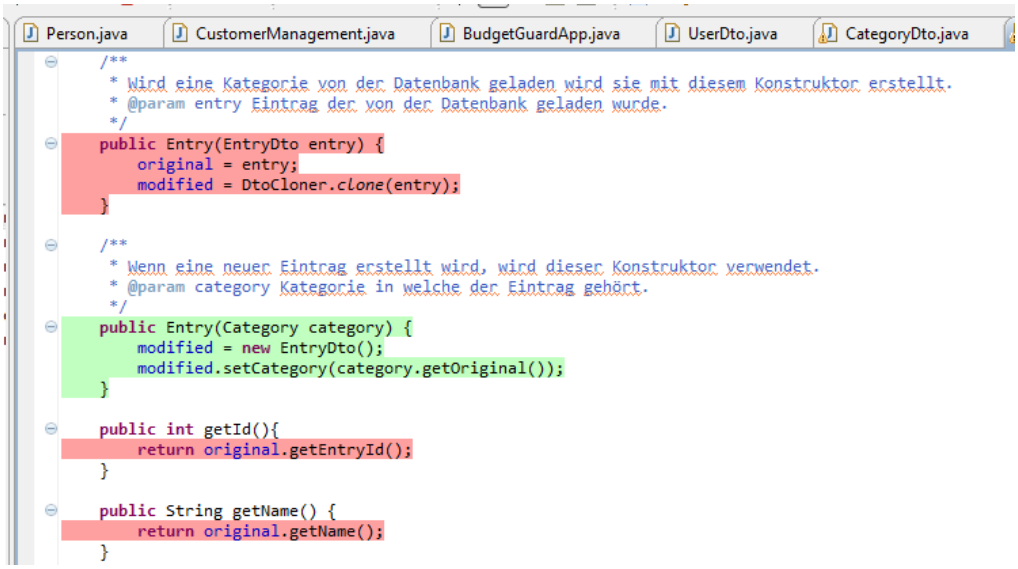


Abbildung 24: EclEmma: Abgedeckte Bereiche

Die roten Markierungen zeigen an, dass diese Methoden nicht getestet werden. Dies vereinfacht die Auswertung erheblich, denn es ist sofort sichtbar, wo noch weiter Tests notwendig sind.

Exports nach XML sind auch möglich, welche zur eventuellen Weiterverarbeitung verwendet werden können.

6.5.3.2.4 Fazit & Empfehlung

Da Test Driven Development den Studenten nahegelegt wird bietet dieses Tool eine optimale Unterstützung dazu. Es ist direkt nach der Ausführung von Test im Coverage Modus ersichtlich, welche Bereiche und Klassen sauber abgedeckt sind und welche nicht. Diese Ansicht eröffnet die Möglichkeit, auf Grund der Abdeckungsraten, gewisse Mängel zu beschreiben und Erklärungen dazu abzugeben.

Deshalb wird EclEmma ebenfalls empfohlen.

6.5.3.3 Checkstyle

6.5.3.3.1 Allgemein

Checkstyle ist ein kostenfreies Tool um diverse Unschönheiten in der Software zu erkennen. Dies sind z.B. zu lange Methoden, keine Kommentare auf public Methoden, leere catch Blöcke, usw. Es werden dem Benutzer Meldungen direkt im Eclipse angezeigt. Weitere Informationen befinden sich unter http://checkstyle.sourceforge.net/index.html.

6.5.3.3.2 Installation

Auch die Installation diese Plug-Ins ist sehr einfach. Folgende Schritte wurden dazu durchgeführt:

1. Starten von Eclipse
2. Über *Help > Eclipse Marketplace..* die Suche nach Checkstyle starten
3. Danach auf *Install* klicken und das Plug-In wird automatisch installiert

Danach ist das Plug-In installiert und kann verwendet werden.

6.5.3.3.3 Benutzbarkeit

Sobald Checkstyle über die Projekteigenschaften aktiviert ist, werden immer direkt im Code Fehlermeldungen angezeigt und die betreffende Stelle wird gelb markiert, was im folgenden Bild ersichtlich ist.

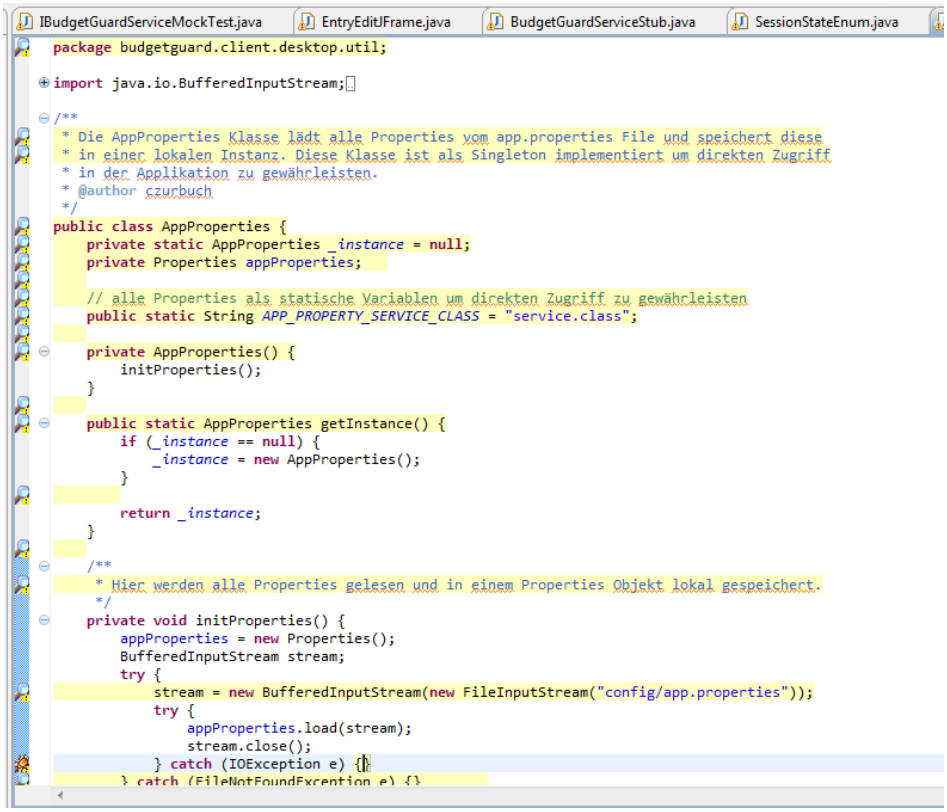


Abbildung 25: Checkstyle: Fehleranzeige

Ein Manko an der Checkstyle Engine ist, dass sehr viele kleine Mängel angezeigt werden und in Kombination mit FindBugs wird es mit der Zeit etwas unübersichtlich.

Gewisse Einschränkungen sind ausserdem sehr unschön, wenn eine Zeile Code z.B. nur 80 Zeilen lang sein darf so wird folgende Zeile Code als Fehler markiert.

```
ry {
   stream = new BufferedInputStream(new FileInputStream("config/app.properties"));
```

Abbildung 26: Checkstyle: Beispiel Fehler

Erst nach mehreren Anpassungen wird folgende Variante akzeptiert von Checkstyle, welche eher unschön aussieht.

```
try {
    stream = new BufferedInputStream(
                new FileInputStream(
                "config/app.properties"));
```

Abbildung 27: Checkstyle: Beispiel Fehler behoben

6.5.3.3.4 Fazit & Empfehlung

Checkstyle überprüft viele Unschönheiten im Code die sehr sinnvoll erscheinen. Jedoch werden extrem viele Styles überprüft, was zu Unübersichtlichkeit führen kann. Diese Einstellungen kann man nicht oder nur sehr mühsam anpassen.

Wir empfehlen es nur den Studenten, welche einen grossen Wert auf extrem sauberen Code legen. Wichtig ist auch, dass das Plug-In von Beginn an der Programmierung verwendet wird um direkt Fehler zu sehen und diese sogleich auch beheben zu können.

6.6 Versionsmanagement

6.6.1 Zweck

Dieses Dokument beschreibt die Beurteilung der beiden Versionsmanagement Tools Git und SVN.

6.6.2 Quellen

Folgend sind die Quellen dieses Dokumentes aufgelistet:

- Anforderungsspezifikation.pdf
- SVN: http://subversion.apache.org/
- Git: http://git-scm.com/
- Vorlesungsunterlagen Git Software-Engineering 2, Stand FS-2011, Marcel Huber

6.6.3 Tools

Bei der Evaluation wurde der vorgängig in der Anforderungsspezifikation definierte Funktionsumfang analysiert und ausgewertet.

6.6.3.1 SVN

6.6.3.1.1 Allgemein

SVN steht für Apache Subversion und ist ein kostenfreies Werkzeug zur Versionskontrolle. SVN ist ein Open Source Projekt das von der Apache Foundation betreut wird. Weitere Informationen finden Sie unter: http://subversion.apache.org/ und http://en.wikipedia.org/wiki/Apache_Subversion

Ein SVN Server wird von der Schule zur Verfügung gestellt, auf welchem man sich ohne Weiteres ein Projekt erstellen kann. Man erhält 250 Megabyte Speicherplatz zugeteilt.

Für den Client ist TortoiseSVN sehr zu empfehlen, erhältlich unter:
http://tortoisesvn.net/downloads.html

Es sind auch diverse andere Tools erhältlich, die wie TurtoiseSVN meist selbsterklärend in der Bedienung sind.

6.6.3.1.2 Analyse & Fazit

SVN ist von der Handhabung her sehr einfach und vor allem bei kleineren Projekten sehr Empfehlenswert. Den bei kleineren Projekten mit einer überschaubaren Anzahl Mitarbeiter sind die Funktionen des Branching und Tagging nicht so wichtig. Die eben genannten Funktionen sind mit SVN nicht ganz einfach zu Realisieren. Ein weiterer Nachteil bei SVN ist, dass stets eine Verbindung zum Server benötigt wird. Ebenfalls dass bei jedem Auschecken das ganze Repository kopiert wird. Was bei grösseren Projekten einige Sekunden dauern kann

6.6.3.2 Git

6.6.3.2.1 Allgemein

Git ist ein kostenfreies Werkzeug zur Versionskontrolle. Weitere Informationen unter:
http://git-scm.com/ und http://en.wikipedia.org/wiki/Git_(software)

Für den Client ist TortoiseGit sehr zu empfehlen, erhältlich unter:
http://code.google.com/p/tortoisegit/

6.6.3.2.2 Analyse & Fazit

Die Handhabung von Git ist etwas kompliziert. Man benötigt etwas Erfahrung im Umgang mit Git.

Vorteile von Git sind, Lokales Branching/Tagging, schneller Austausch zwischen den Clients, mehrere Versionsbäume. Git ist optimal für grössere Projekte mit mehreren Mitarbeitern.

6.6.4 Gegenüberstellung

Auf eine Gegenüberstellung im direkten Sinn wurde in diesem Fall verzichtet, da beide Tools Bestandteil der SE –Vorlesungen sind. Im Projektmanagement Tool kann sowohl Git wie auch SVN eingebunden werden. Installation und Einrichtungshinweise werden für beide erstellt.

6.7 Projektmanagement Tools

6.7.1 Zweck

Dieses Dokument beschreibt die Evaluation und anschliessende Auswahl eines Projektmanagement Tools. Es soll aufzeigen aus welchen Gründen eine entsprechende Wahl getroffen wurde.

6.7.2 Quellen

Folgend sind die Quellen dieses Dokumentes aufgelistet:

- Anforderungsspezifikation.pdf
- Redmine (http://www.redmine.org/)
- Trac (http://trac.edgewall.org/)

6.7.3 Tools

Es wurden mehrere verschiedene Projektmanagement Tools angesehen und die Auswahl zur Evaluation wurde auf zwei beschränkt. Es sind dies Redmine und Trac. Es gibt viele weitere Tools mit Cloud-Service, welche zwar über einen geringeren Konfigurationsaufwand verfügen, aber in der Gratisversion nur sehr beschränkte Funktionalitäten bieten. Deshalb wurde von Cloud-Lösungen im Allgemeinen abgesehen.

Bei der Evaluation wurde der vorgängig in der Anforderungsspezifikation definierte Funktionsumfang analysiert und ausgewertet.

6.7.3.1 Redmine

6.7.3.1.1 Allgemein

Redmine ist eine Open-Source Lösung und somit frei erhältlich. Es ist für die gängigen Linux-Variante (z.B. Fedora, Ubuntu), Windows und Mac iOS verfügbar. Dies rührt daher, dass Redmine mit Ruby geschrieben wurde.

6.7.3.1.2 Installation

Die Installation von Redmine gestaltet sich als nicht ganz einfach.

1. Ruby, Rails, Rake und die dazugehörigen Pakete zur Datenbankanbindung
2. Eine Datenbank wie PostgreSQL oder MySQL muss eingerichtet werden
3. Ein Apache2 Server installieren
4. Redmine installieren
5. Datenbank einbinden

Die Redmine Version verlangt explizit nach einer Ruby -Version. Das heisst Redmine funktioniert beispielsweise nur mit Ruby Version 1.8 jedoch nicht mit Version 1.9. Dies verursacht Probleme bei der Installation.

Alternativ kann auch ein One-Click-Installer eines Drittanbieters verwendet werden. Der Redmine Stack von BitNami ist zu empfehlen. Ist ebenfalls kostenfrei, jedoch handelt es sich darin nicht um die aktuellste Version von Redmine. Für einen Server ohne grafische Oberfläche ist der Installer jedoch ungeeignet. Bei BitNami kann auch ein Cloud-Service bezogen werden, dies wurde jedoch nicht weiter beachtet.

6.7.3.1.3 Benutzbarkeit

Redmine ist optisch sehr angenehm. Es bietet sehr viele verschiedene Funktionen, ist deshalb etwas komplex und benötigt eine längere Einarbeitung. Die Erstellung eines Benutzers oder Projekts ist sehr einfach und kann direkt im Webinterface ausgeführt werden.

Es verschiedene Views wie Timeline, Roadmap und Tickets. Diese werden in den drei folgenden Abschnitten beschrieben. Auf der Timeline können zudem verschieden Filter angewendet werden.

Des Weiteren sind Graphen zur Veranschaulichung des Projektfortschritts verfügbar. Da das Repository direkt in Redmine eingebaut werden kann, werden alle eingecheckten Dateien im Webbrowser angezeigt.

6.7.3.1.3.1 Timeline

Auf dieser Ansicht sind laufende Änderungen sichtbar. Es wird angezeigt wer was wann gemacht hat (z.B. ein Dokument eingecheckt hat). Es kann nach verschiedenen Kriterien gefiltert werden (Milestones, Datum, usw.).

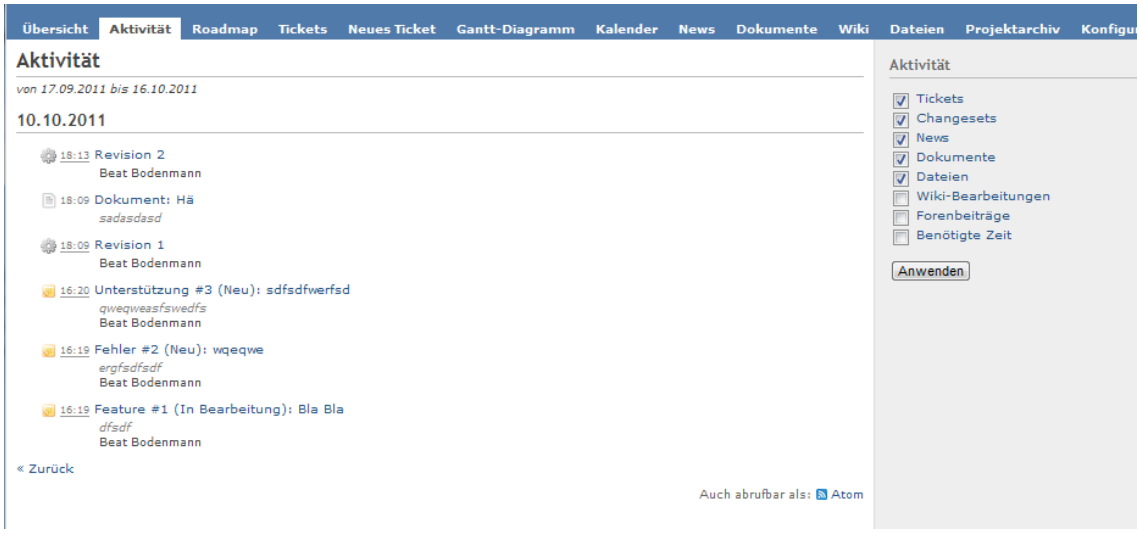


Abbildung 28: Evaluation Projektmanagement Tools: Redmine: Timeline

6.7.3.1.3.2 Roadmap

In dieser Ansicht ist der Fortschritt eines Projekts ersichtlich, man sieht den Fortschritt der einzelnen Versionen. Ausserdem sind die Daten der Versionen, Anzahl Tickets (offen/geschlossen) zu sehen.

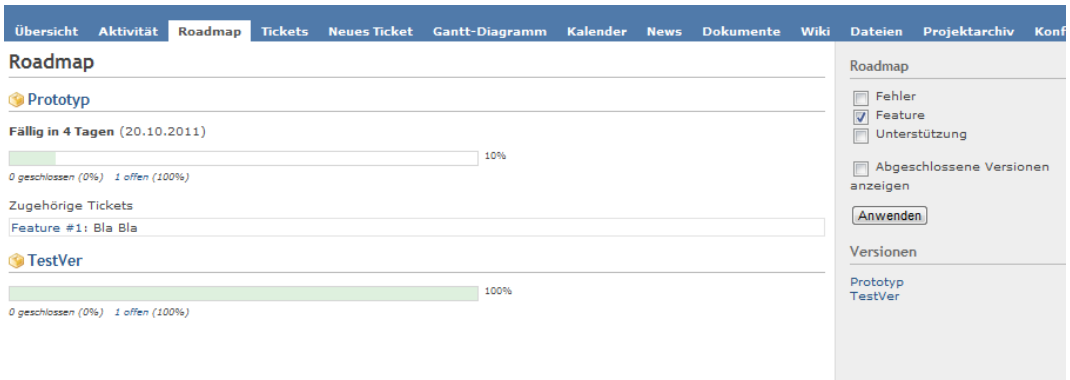


Abbildung 29: Evaluation Projektmanagement Tools: Redmine: Roadmap

6.7.3.1.3.3 Tickets

Zeigt alle Tickets an, die gemäss Filter ausgewählt wurden.

Abbildung 30: Evaluation Projektmanagement Tools: Redmine: Tickets

6.7.3.1.4 Funktionalität

Positiv:

- Einbindung des Git/SVN – Repository möglich
- Projekte erstellen sehr einfach
- Unterprojekte innerhalb eines Projekts
- Kategorie
- Tickets (können Projekt, Kategorie, Priorität und Version zugeordnet werden).
- Firmen/Team Struktur kann abgebildet werden da jedes Teammitglied erfasst und seine Funktion definiert werden kann, sowie die Rechte definiert werden können.
- Jeder Benutzer kann Zeiten auf den Tickets buchen, welche er bearbeitet hat.
- Exportfunktionen sind vorhanden.
- Integrierter Team Kalender

Negativ:

- Es können keine Meilensteine erfasst werden, welchen Tickets zugeordnet werden können (Dies kann mit Versionen realisiert werden, jedoch Nachteile in der Auswertung)

6.7.3.2 Trac

6.7.3.2.1 Allgemein

Trac ist ebenfalls kostenfrei. Es ist mit Python aufgebaut und kann auf den gängigen Plattformen eingesetzt werden.

6.7.3.2.2 Installation

Anfänglich gestaltete sich die Installation als einfacher verglichen mit Redmine. Bei der Datenbank-Einbindung stösst man jedoch auf mehrere Probleme.

1. Installiere Datenbank
2. Installiere Python
3. Installiere Trac
4. Einbinden der Datenbank

Für Trac muss vorgängig ein Admin User und Projekt in der Konsole erstellt werden.

Auch für Trac wird von der Firma BitNami einen One-Click-Installer angeboten.

6.7.3.2.3 Benutzbarkeit

Trac überzeugt mit einem einfachen und minimalistischem Design. Bietet jedoch, verglichen mit Redmine, weniger Funktionalitäten. Dadurch hat man einen schnellen Überblick über die Möglichkeit dieses Tools. Bietet verschiedene Views wie Timeline, Roadmap und Tickets. Auf der Timeline können zudem verschiedene Filter angewendet werden. Die Ticketübersicht bietet verschiedene Arten von Reports, die eine schnelle Suche nach dem gewünschten Ticket ermöglichen. Die drei Views werden in den folgenden Abschnitten noch beschrieben.

Die Zeiterfassung wurde als Plug-In nach installiert, lässt jedoch zu wünschen übrig. Man kann zwar Soll-und Ist-Zeiten erfassen, es werden einem jedoch keine übersichtlichen Statistiken oder Graphen geboten.

6.7.3.2.3.1 Timeline

Auf dieser Ansicht sind laufende Änderungen sichtbar. Es wird angezeigt wer was wann gemacht hat (z.B. ein Dokument eingecheckt hat). Es kann nach verschiedenen Kriterien gefiltert werden (Meilensteine, Datum, usw.).

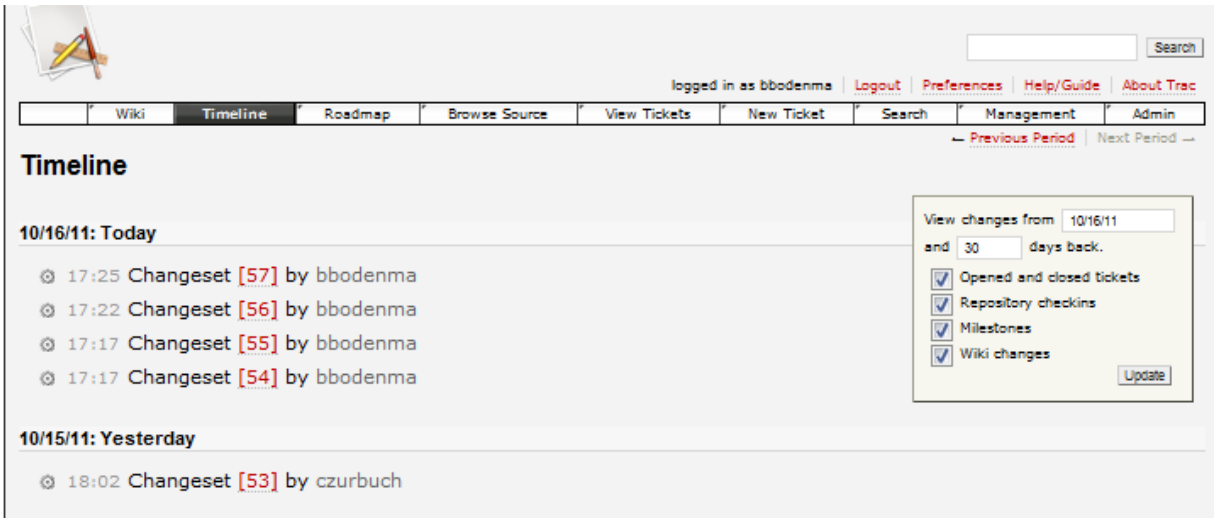


Abbildung 31: Evaluation Projektmanagement Tools: Trac: Timeline

6.7.3.2.3.2 Roadmap

In dieser Ansicht ist der Fortschritt eines Projekts ersichtlich, man sieht den Fortschritt der einzelnen Meilensteine. Ausserdem sind die Daten der Meilensteine, Anzahl Tickets (offen/geschlossen) zu sehen.

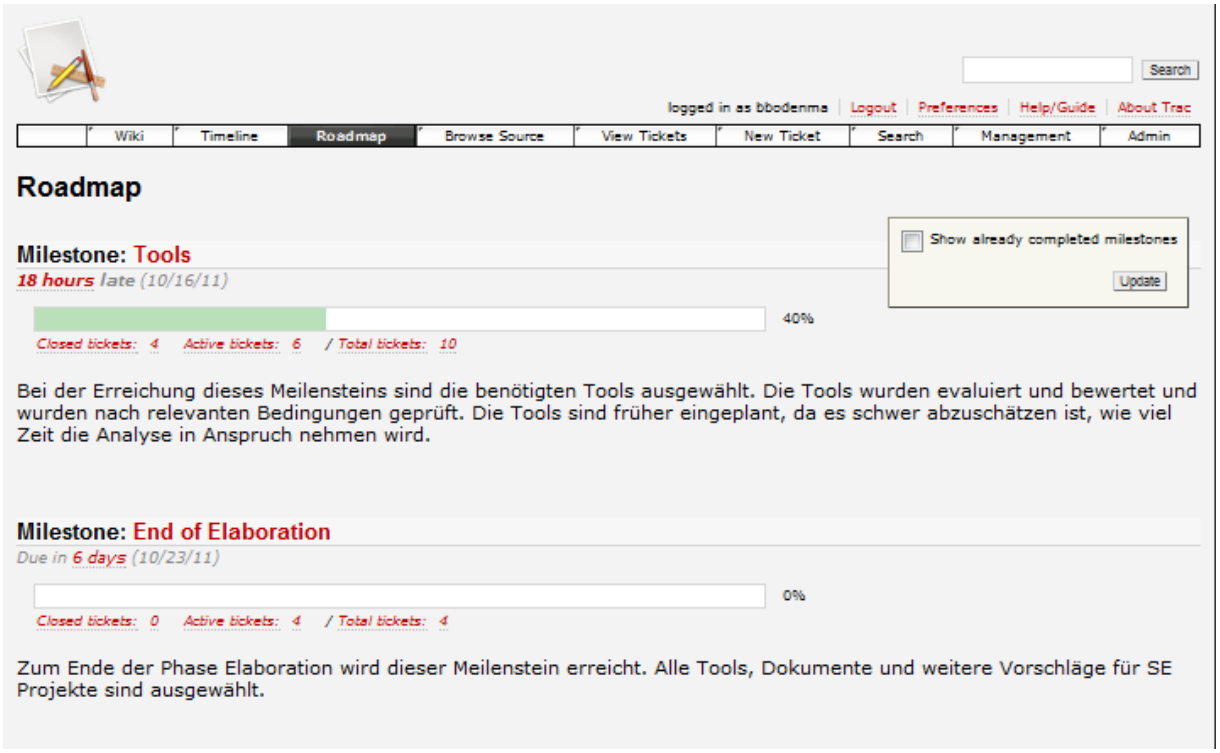


Abbildung 32: Trac Roadmap

6.7.3.2.3.3 Tickets

Zuerst kann der gewünschte Report ausgewählt werden:

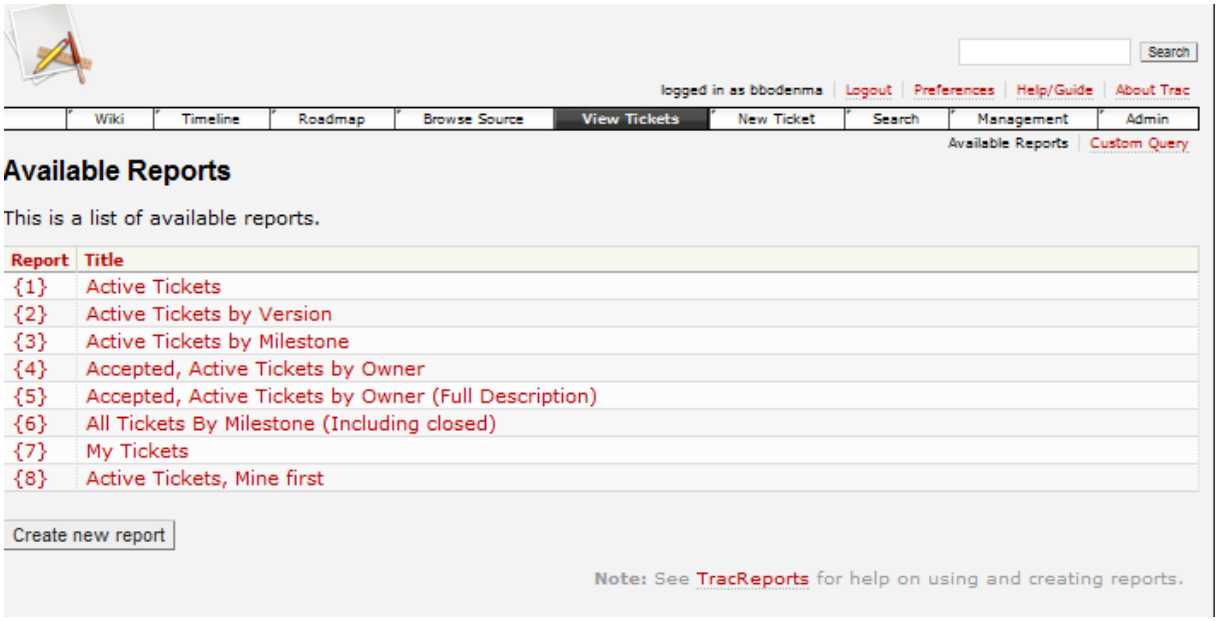
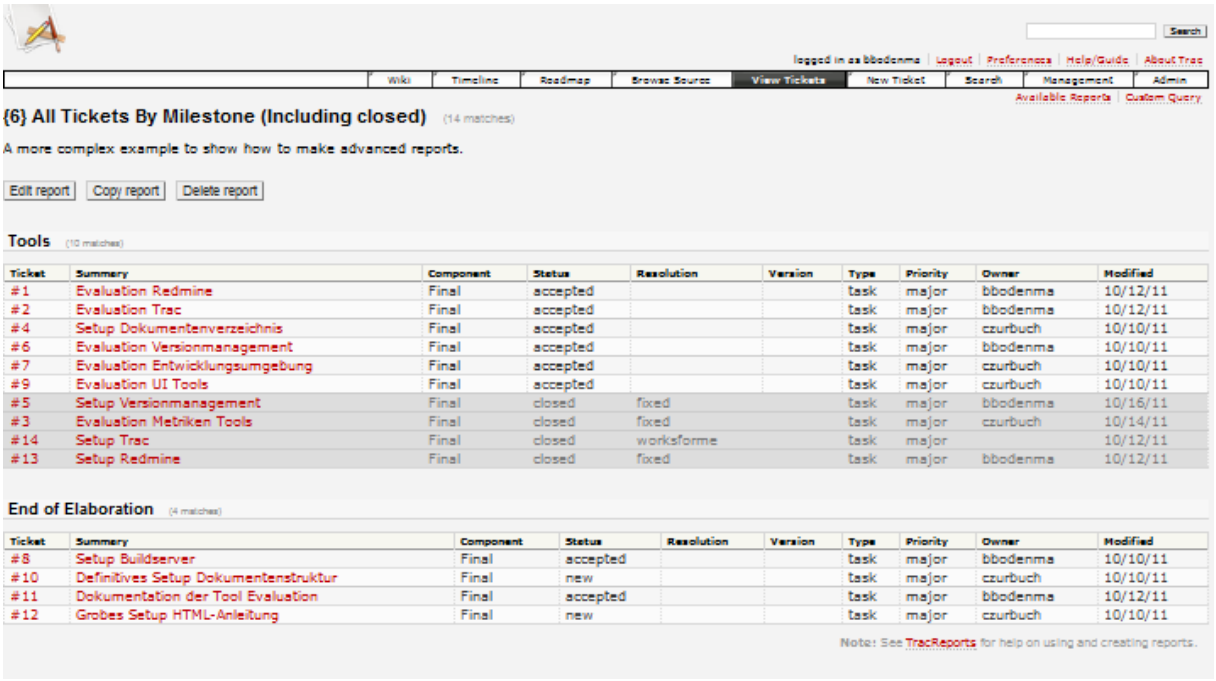


Abbildung 33: Evaluation Projektmanagement Tools: Trac: Ticket Reports

Danach erhält die entsprechende Ansicht. Zum Beispiel:



{6} All Tickets By Milestone (Including closed) (14 matches)

A more complex example to show how to make advanced reports.

Edit report | Copy report | Delete report

Tools (10 matches)

Ticket	Summary	Component	Status	Resolution	Version	Type	Priority	Owner	Modified
#1	Evaluation Redmine	Final	accepted			task	major	bbodenma	10/12/11
#2	Evaluation Trac	Final	accepted			task	major	bbodenma	10/12/11
#4	Setup Dokumentenverzeichnis	Final	accepted			task	major	czurbuch	10/10/11
#6	Evaluation Versionmanagement	Final	accepted			task	major	bbodenma	10/10/11
#7	Evaluation Entwicklungsumgebung	Final	accepted			task	major	czurbuch	10/10/11
#9	Evaluation UI Tools	Final	accepted			task	major	czurbuch	10/10/11
#5	Setup Versionmanagement	Final	closed	fixed		task	major	bbodenma	10/16/11
#3	Evaluation Metriken Tools	Final	closed	fixed		task	major	czurbuch	10/14/11
#14	Setup Trac	Final	closed	worksforme		task	major		10/12/11
#13	Setup Redmine	Final	closed	fixed		task	major	bbodenma	10/12/11

End of Elaboration (4 matches)

Ticket	Summary	Component	Status	Resolution	Version	Type	Priority	Owner	Modified
#8	Setup Buildserver	Final	accepted			task	major	bbodenma	10/10/11
#10	Definitives Setup Dokumentenstruktur	Final	new			task	major	czurbuch	10/10/11
#11	Dokumentation der Tool Evaluation	Final	accepted			task	major	bbodenma	10/12/11
#12	Grobes Setup HTML-Anleitung	Final	new			task	major	czurbuch	10/10/11

Note: See TracReports for help on using and creating reports.

Abbildung 34: Evaluation Projektmanagement Tools: Trac: Ticket Übersicht

Es wurde “All Tickets By Milestone” gewählt, auf dieser Ansicht werde alle Tickets mit Status, Besitzer und weiter Informationen geordnet nach Meilenstein dargestellt.

6.7.3.2.4 Funktionalität

Positiv:

- Meilensteine erstellbar.
- Tickets erstellen, welche einem Projekt und einem Meilenstein zugeordnet werden können.
- User mit entsprechenden Rechten erstellen (lesen, editieren oder Admin-User).
- Beschränkte Exportfunktionen sind vorhanden.

Negativ:

- Admin muss vorgängig via Konsole erstellt werden.
- Neue Projekte müssen via Konsole initiiert werden.
- Standardmässig keine Zeiterfassung, Plug-In kann Installiert werden.

6.7.4 Gegenüberstellung

6.7.4.1 Bewertungspunkte

In den folgenden Abschnitten werden die einzelnen Bewertungspunkte beschrieben. Im abschliessenden Abschnitt erfolgt die Bewertung der einzelnen Tools bezüglich der Bewertungspunkte. Jeder Bewertungspunkt beinhaltet noch eine Gewichtung von 1 bis 3 je nach Wichtigkeit.

6.7.4.1.1 Projekterstellung

Ein neues Projekt kann einfach und schnell erstellt werden.

6.7.4.1.2 Benutzererstellung

Ein neuer Benutzer kann einfach und schnell erstellt werden.

6.7.4.1.3 Meilensteine

Meilensteine sind erfassbar mit einem Datum erfasst werden. Damit Auswertungen über die Meilensteine gemacht werden können.

6.7.4.1.4 Ticketing

Es ist möglich Tickets zu erfassen und diese, einem Projektmitglied zuzuordnen. Die Tickets sollten ausserdem einem Meilenstein zugeteilt werden können. Ausserdem sollten Tickets Kategorien zuteilbar sein um z.B. Auswertung über den Zeitaufwand im Projektmanagement zu machen.

6.7.4.1.5 Zeitmanagement

Für Meilensteine und Tickets können Soll-Zeiten erfasst werden.

6.7.4.1.6 Zeiterfassung

Auf den Tickets können die Zeiten verbucht werden. Diese Zeiten sollten auch leicht auswertbar sein, damit man eine einfache und wöchentliche Übersicht über die Tätigkeiten der einzelnen Projektmitarbeiter erhält.

6.7.4.1.7 Export

Die eingegebenen Daten können exportiert werden in anständigen Formaten wie z.B. Excel oder PDF.

6.7.4.2 Übersicht

		Redmine			Trac		
	Gew.	**Wert.**	**Beschreibung**	**Total**	**Wert.**	**Beschreibung**	**Total**
2.1.1 Projekt-erstellung	1	10	Ist über das Webinterface einfach zu erstellen.	10	4	Ist über die Konsole zu erstellen.	4
2.1.2 Benutzer-erstellung	1	10	Ist via Webinterface möglich, einfach selbsterklärend.	10	8	Der Admin User muss vorgängig in der Konsole erstellt werden, danach einfach via Webinterface.	8
2.1.3 Meilensteine	2	8	Man kann Version erfassen, können gleichbehandelt werden.	16	10	Meilensteine sind einfach erfassbar.	20
2.1.4 Ticketing	3	8	Tickets sind einfach und schnell erfassbar, und können zugeordnet werden.	24	8	Tickets sind einfach und schnell erfassbar und können zugeordnet werden.	24
2.1.5 Zeitmana-gement	3	8	Zeitschätzungen können Versionen, Projekten und Tickets hinzugefügt werden.	24	6	Nur für Tickets können Soll-Zeiten erfasst werden und sehr unübersichtlich. Ausserdem nur als Plug-In verfügbar.	18
2.1.6 Zeiterfassung	3	8	Auf den Tickets können Zeiten gebucht werden, dies ist danach ersichtlich.	24	7	Auf den Tickets können Zeiten gebucht werden, welche danach ausgewertet werden können.	21
2.1.7 Export	1	10	Je nach View sind CSV und/oder PDF Export verfügbar.	10	3	Teilweise ist ein CSV Export verfügbar.	3
TOTAL				**118**			**98**

Tabelle 8: Evaluation Projektmanagement Tools: Gegenüberstellung

6.7.4.3 Auswahl

Redmine ist zwar etwas komplexer und benötigt mehr Einarbeitungszeit, bietet dafür aber auch mehr Funktionen. Die Zeiterfassung von Redmine ist erheblich besser, ebenfalls überzeugte Redmine bei den Exportfunktionen. Deshalb wird Redemine empfohlen.

6.8 Jenkins

6.8.1 Zweck

Dieses Dokument beschreibt die Evaluation der Einsetzbarkeit eines Jenkins Build-Server für ein SE-2 Projekt. Es werden ausserdem die Plug-Ins: JUnit, Emma und FindBugs evaluiert.

6.8.2 Quellen

Folgend sind die Quellen dieses Dokumentes aufgelistet:

- Anforderungsspezifikation.pdf
- SE-2 Vorlesungsunterlagen, Projektautomation, Stand FS-2011, Marcel Huber
- Jenkins Homepage: http://jenkins-ci.org/
- http://de.wikipedia.org/wiki/Jenkins_(Software)
- Emma: http://emma.sourceforge.net/userguide/userguide.html

6.8.3 Tools

6.8.3.1 Jenkins

6.8.3.1.1 Allgemein

Jenkins ist einer der meistverbreiteten Build Servern, denn er ist webbasiert und damit plattformunabhängig für den Benutzer beliebig erweiterbar. Jenkins wurde zuerst von Sun als Hudson entwickelt. Nach der Übernahme von Oracle, ging das Namensrecht für Hudson an Oracle. Daraufhin wurde aus Hudson Jenkins. Jenkins ist ein Open-Source Projekt und somit frei erhältlich.

6.8.3.1.2 Installation

Die Installation eines Jenkins Build Server ist unter Ubuntu wie folgt möglich:

7. Jenkins Debian Package Key zum System hinzufügen
   wget -q -O - http://pkg.jenkins-ci.org/debian/jenkins-ci.org.key | sudo apt-key add -
8. Jenkins binary zur sources.list hinzufügen
   deb http://pkg.jenkins-ci.org/debian binary/
9. *apt-get update*
10. *apt-get install jenkins*
11. Falls noch benötigt einen Apache Server Installieren.

Danach ist Jenkins via http://ihrserver.com:8080 erreichbar.

Quelle: http://pkg.jenkins-ci.org/debian/

6.8.3.1.3 Benutzbarkeit

6.8.3.1.3.1 Startseite

Jenkins besitzt eine Schlanke Übersichtsseite, auf welcher jedes Projekt mit dem aktuellen Zustand sofort ersichtlich ist.

Abbildung 35: Evaluation Jenkins: Startseite

6.8.3.1.3.2 Neuer Job

Ein Projekt, bei welchem man eine kontinuierliche Integration wünscht, muss als Job angelegt werden. Jenkins bietet unter anderem folgende Job-Arten: Free-Style Projekt und Maven Projekt.

Beim Free-Style Projekt können alle Einstellungen beliebig vorgenommen werden, beim Maven Projekt hingegen sind gewisse Teile für den Einsatz mit Maven optimiert. Da in SE-2 Ant unterrichtet wird, wurde auf die Analyse von Maven und somit des Maven Projekt verzichtet.

Hinweise zur Erstellung eines Free-Style Projektes sind der Anleitung zu entnehmen.

6.8.3.1.3.3 Jenkins Verwalten

Abbildung 36: Evaluation Jenkins: Verwaltungsseite

In der Verwaltung können unter anderem die Systemeinstellungen geändert und Plug-Ins installiert werden. Die Liste der verfügbaren Plug-Ins ist riesig, Emma und FindBugs wurden installiert und genauer angeschaut.

6.8.3.1.4 Fazit

Jenkins bietet eine grosse Palette an Funktionalitäten. Der Umgang mit Ant ist manchmal etwas schwierig und benötigt etwas Übung. Ein Build Server ist bei grösseren Teams und Projekten unerlässlich, denn er ermöglicht eine genaue Auswertung verschiedener Arbeitsschritten und bietet eine einheitliche Testumgebung sowie einheitliche Builds. Werden letztere Lokal ausgeführt, sind immer Lokale Einstellungen des jeweiligen Users entscheidend. Dies kann Fehler verursachen. Um dies zu vermeiden ist die Einsetzung eines Build-Servers empfohlen.

6.8.3.2 Emma

6.8.3.2.1 Allgemein

Emma ist ein Tool welches die Testabdeckung des Codes analysiert. Es ist als Eclipse und als Jenkins Plug-In verfügbar. Auf dem Jenkins-Server bietet es ein Instrument der Kontrolle, ob der Code auch getestet wird.

6.8.3.2.2 Installation

Das Plug-In kann im Jenkins-Plug-In Manager heruntergeladen und installiert werden. Danach muss Jenkins neu gestartet werden. Zusätzlich muss noch die Emma Library heruntergeladen werden, welche später im Ant-File eingebunden werden muss.

Library Download unter: http://emma.sourceforge.net/downloads.html

6.8.3.2.3 Benutzbarkeit

Bei einem klassischen Projekt Setup tritt als einziges Problem die Einbindung mit Ant auf. Diese ist sehr aufwändig. Zur Einrichtung sind mehrere Schritte notwendig. Zuerst muss der Source Code kompiliert werden, danach wird dieser instrumentiert, wodurch Emma die benötigten Metadaten sammelt. Anschliessend müssen die Tests durchlaufen werden, wobei die Abdeckung berechnet wird.

Aus den Metadaten und Abdeckung wird dann ein XML Report generiert welcher von Jenkins eingelesen und ausgewertet wird.

```
<!--Run Emma -->
<target name="emma" depends="compileTest, compileSrc">
<echo>Doing Emma instrumented at: ${emma.inst.dir} </echo>
    <emma enabled="true">
        <instr instrpath="${emma.prod.dir}" destdir="${emma.inst.dir}"
            metadatafile="${emma.coverage.dir}/metadata.emma" merge="true"/>
    </emma>
<path id="emma.classpath">
    <pathelement location="${emma.inst.dir}"/>
    <path refid="project.classpath"/>
    <path refid="emma.lib"/>
</path>
<echo>RunTest for Emma</echo>
<junit haltonerror="no" haltonfailure="no" showoutput="yes" printsummary="yes" filtertrace="no" fork="yes">
    <jvmarg value="-Demma.coverage.out.file=${emma.coverage.dir}/coverage.emma" />
    <jvmarg value="-Demma.coverage.out.merge=true" />
    <classpath refid="emma.classpath" />
    <formatter type="plain"/>
    <formatter type="xml" />
    <batchtest todir="${emma.testreport.dir}" filtertrace="no">
        <fileset dir="${emma.test.dir}" includes="**/*Test.class" />
    </batchtest>
    </junit>
<echo>Create Emma Report</echo>
    <emma enabled="true">
        <report>
            <infileset dir="${emma.coverage.dir}" includes="*.emma" />
            <xml outfile="${emma.coverage.dir}/coverage.xml" />
        </report>
    </emma>
</target>
```

Abbildung 37: Evaluation Jenkins: Emma: Antfile (Auszug 1)

Man kann sich mit einer Test-Runner-Class behelfen, diese kann danach von emmajava aufgerufen werden. Emma Java durchläuft den ganzen Code und sammelt die relevanten Abdeckungsdaten, diese werden danach ebenfalls als XML ausgegeben. Dadurch wird es wesentlich einfacher. Man verliert jedoch gewisse Filtering-Funktionen.

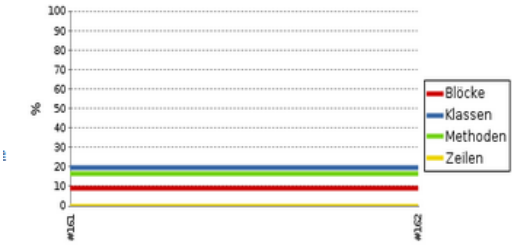
```
<!-- Run Emma -->
<target name="emma" depends="compileTest, compileSrc">

<emmajava enabled="true" fork="yes" libclasspathref="emma.lib"
          fullmetadata="yes" sourcepath="${src.dir}"
          classname="RunAllTests"
          classpathref="project.classpath" >
  <xml outfile="${emma.coverage.dir}/coverage.xml" />
</emmajava>
</target>
```

Abbildung 38: Evaluation Jenkins: Emma: Antfile (Auszug 2)

6.8.3.2.4 Auswertung

Das Plug-In bietet zur Auswertungen eine Momentaufnahme und eine Übersicht der Entwicklung über alle Builds.

Emma Coverage Report



Overall Coverage Summary

Name	Klassen	Methoden	Blöcke
all classes	19.5% 17/87	16.6% 97/583	8.7% 1636/18704

Coverage Breakdown by Package

Name	Klassen	Methoden	Blöcke
budgetguard.client.desktop.bll	50.0% 1/2	8.7% 2/23	1.2% 5/420
budgetguard.client.desktop.bll.domain	80.0% 4/5	40.6% 28/69	50.1% 369/737
budgetguard.client.desktop.bll.domain.util	50.0% 2/4	27.3% 3/11	56.8% 42/74
budgetguard.client.desktop.bll.exception	0.0% 0/1	0.0% 0/1	0.0% 0/3
budgetguard.client.desktop.dal	25.0% 1/4	16.0% 12/75	6.9% 308/4447
budgetguard.client.desktop.dal.dto	100.0% 4/4	63.9% 39/61	51.5% 709/1376
budgetguard.client.desktop.dal.exception	40.0% 2/5	10.0% 4/40	9.4% 46/490
budgetguard.client.desktop.dal.shared	50.0% 1/2	12.5% 3/24	26.2% 55/210
budgetguard.client.desktop.util	50.0% 2/4	42.9% 6/14	36.7% 102/278
budgetguard.client.desktop.view	0.0% 0/42	0.0% 0/215	0.0% 0/9903
budgetguard.client.desktop.view.draft	0.0% 0/7	0.0% 0/18	0.0% 0/314
budgetguard.client.desktop.view.item	0.0% 0/1	0.0% 0/6	0.0% 0/37
budgetguard.client.desktop.view.util	0.0% 0/6	0.0% 0/26	0.0% 0/415

Abbildung 39: Evaluation Jenkins: Auswertung der Emma Coverage-Analyse

6.8.3.2.5 Fazit

Emma für Jenkins ist ein gutes Werkzeug und bei der Verwendung eines Buildservers einsetzbar. Es ermöglicht eine Verfolgung der Entwicklung der Test Abdeckung eines Projekts. Wird ein Jenkins-Build-Server eingesetzt, ist es empfohlen dafür auch Emma einzusetzen.

6.8.3.3 FindBugs

6.8.3.3.1 Allgemein

FindBugs ist ein Tool, welches den Code analysiert und auf Unschönheiten und mögliche Fehler hinweist. Es ist als Eclipse und als Jenkins Plug-In verfügbar. Auf dem Jenkins-Server bietet es ein Instrument der Kontrolle und zur Erstellung der Statistik.

6.8.3.3.2 Installation

Das Plug-In kann im Jenkins-Plug-In Manager heruntergeladen und installiert werden. Danach muss Jenkins neu gestartet werden. Zusätzlich muss noch die FindBugs Library heruntergeladen welche später im Ant-File eingebunden werden muss.

Library Download unter: http://findbugs.sourceforge.net/downloads.html

6.8.3.3.3 Benutzbarkeit

FindBugs ist fähig direkt JAR-Dateien zu analysieren. Dies ermöglicht eine einfache Ausführung.

Zuerst FindBugs Pfad und Task definieren, danach JAR-Datei erstellen und die darin erhaltenen Files analysieren.

```
<!-- FindBugs einbinden -->
<path id="findbugs.lib">
        <fileset dir="${findbugs.dir}/lib">
            <include name="*.jar"/>
        </fileset>
</path>
<taskdef name="findbugs" classname="edu.umd.cs.findbugs.anttask.FindBugsTask">
 <classpath refid="findbugs.lib" />
</taskdef>
```

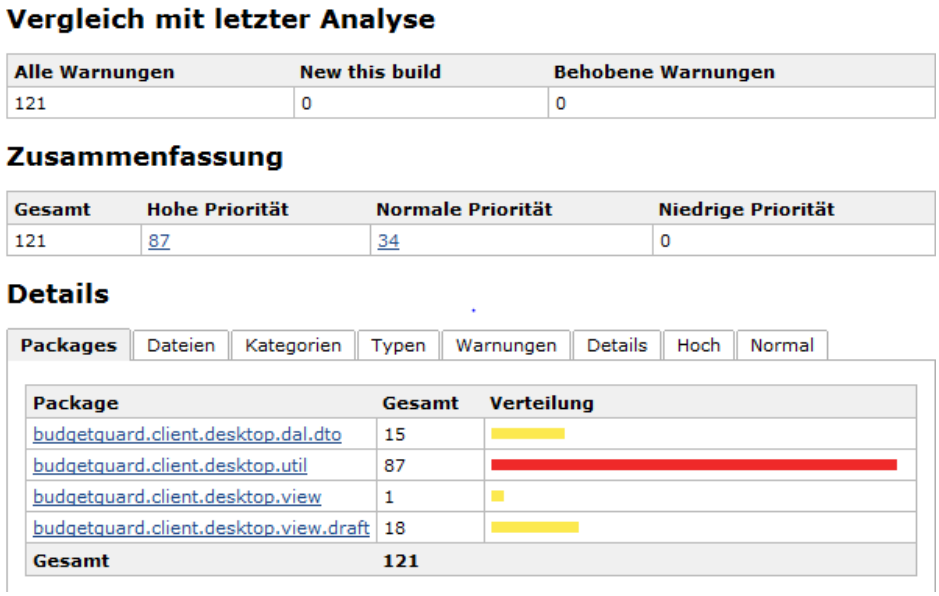
Abbildung 40: Evaluation Jenkins: FindBugs: Antfile (Auszug 1)

```
  <!-- find Bugs -->
  <target name="findBugs">
  <findbugs home="${findbugs.dir}"
            output="xml"
            outputFile="${build.report.bugs.dir}/bugs.xml" >
    <sourcePath path="${src.dir}" />
    <class location="${dist.dir}/${jar.name}" />
  </findbugs>
</target>
```

Abbildung 41: Evaluation Jenkins: FindBugs: Antfile (Auszug 2)

6.8.3.3.4 Auswertung

Das Plug-In bietet zur Auswertung eine Momentaufnahme und die Entwicklung des Codes gegenüber der letzten Auswertung.



Vergleich mit letzter Analyse

Alle Warnungen	New this build	Behobene Warnungen
121	0	0

Zusammenfassung

Gesamt	Hohe Priorität	Normale Priorität	Niedrige Priorität
121	87	34	0

Details

Packages | Dateien | Kategorien | Typen | Warnungen | Details | Hoch | Normal

Package	Gesamt	Verteilung
budgetguard.client.desktop.dal.dto	15	
budgetguard.client.desktop.util	87	
budgetguard.client.desktop.view	1	
budgetguard.client.desktop.view.draft	18	
Gesamt	**121**	

Abbildung 42: Evaluation Jenkins: FindBugs Auswertung Jenkins

6.8.3.3.5 Fazit

FindBugs für Jenkins ist ein gutes Werkzeug und bei der Verwendung eines Buildservers einsetzbar. Es ermöglicht eine Verfolgung der Entwicklung des Codes und dessen Qualität innerhalb eines Projekts. Wird ein Jenkins-Build-Server eingesetzt, ist es empfohlen dafür auch FindBugs einzusetzen, zusätzlich zu FindBugs in Eclipse.

7. Anleitungen

7.1 Eclipse Metrics

7.1.1 Installation

1) Starten Sie Eclipse.

2) *Eclipse Marketplace* starten.

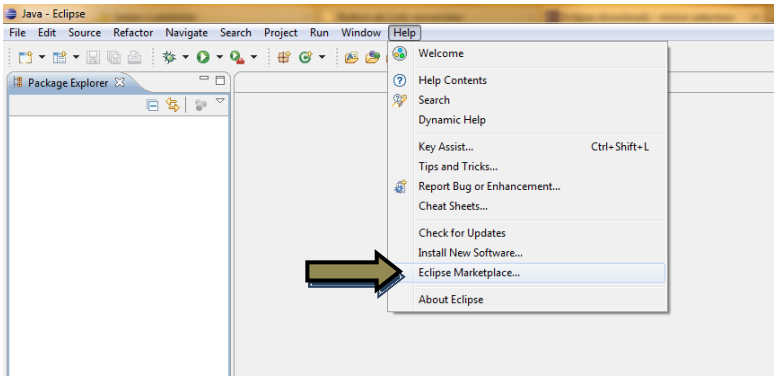


Abbildung 43: Anleitung Eclipse Metrics: Eclipse Hilfemenü

3) Suche nach „Eclipse Metrics“ und anschliessend auf *Install* klicken.

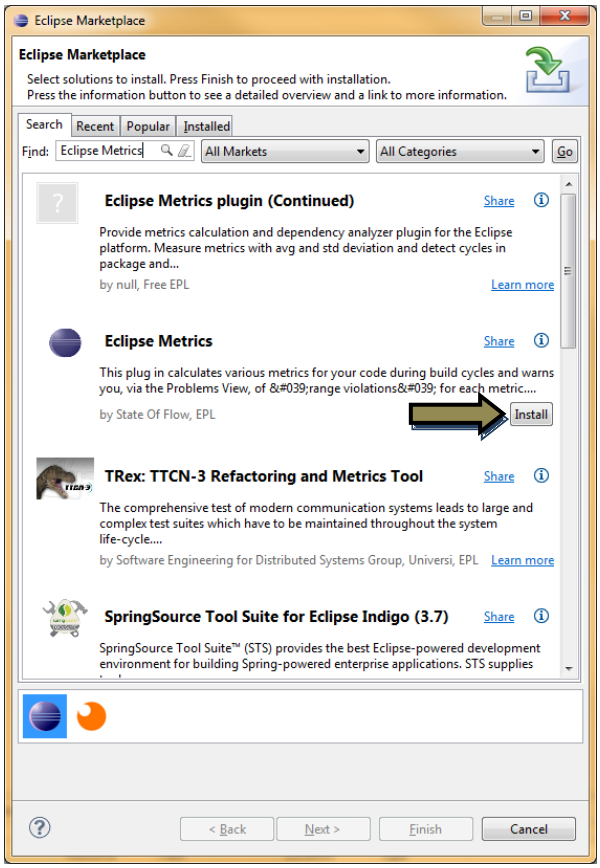


Abbildung 44: Anleitung Eclipse Metrics: Eclipse Marketplace

Es kann einige Minuten dauern, bis alle Pakete geladen sind.

4) Stellen Sie sicher, dass alle Elemente ausgewählt sind und klicken Sie auf *Next*.

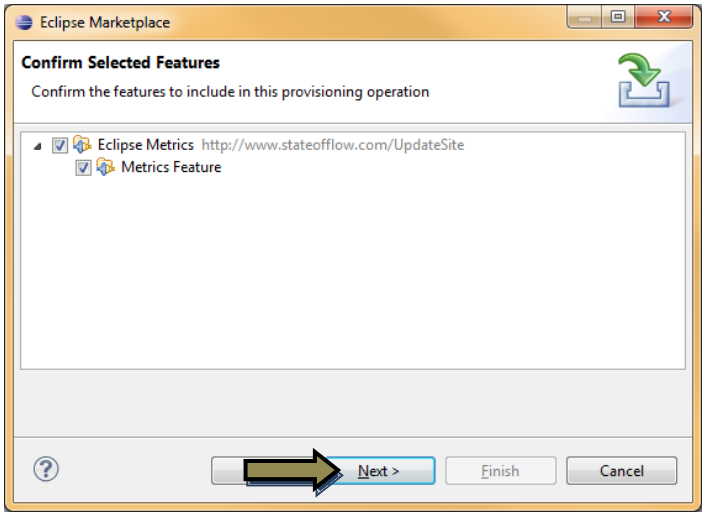


Abbildung 45: Anleitung Eclipse Metrics: Download

5) Akzeptieren Sie die Lizenzvereinbarung und klicken Sie auf *Finish*.

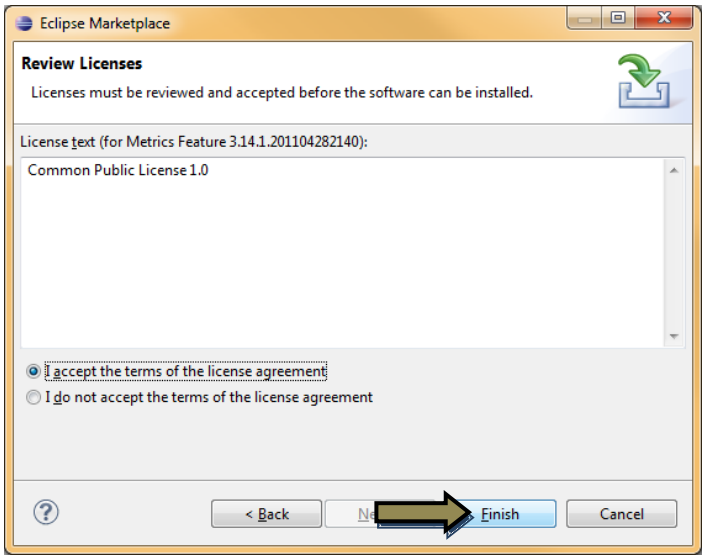


Abbildung 46: Anleitung Eclipse Metrics: Lizenzvereinbarung

6) Das Plug-In wird installiert und kann nach einem Neustart von Eclipse verwendet werden.

Es kann sein das während der Installation eine Warnung auftaucht, diese kann jedoch ignoriert werden. Klicken sie auf *OK* und die Installation wird fortgesetzt.

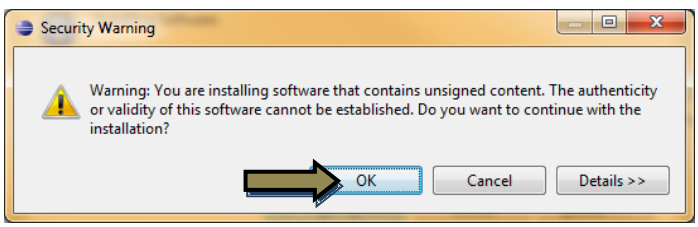


Abbildung 47: Anleitung Eclipse Metrics: Unsigned Content Warnung

7.1.2 Setup

1) Starten Sie Eclipse.

2) Wählen Sie links das gewünschte Projekt aus und klicken Sie auf *Properties*.

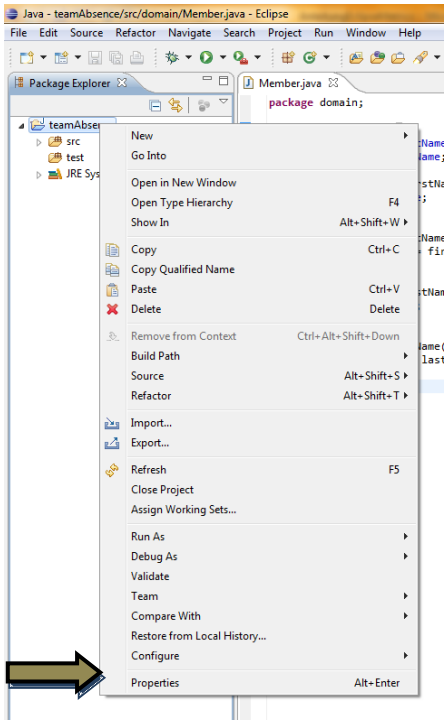


Abbildung 48: Anleitung Eclipse Metrics: Projekt Kontextmenü

3) Wählen Sie den Punkt *Metrics*. Anschliessend setzen Sie einen Haken bei *Enable Metrics Gathering* und klicken Sie auf *OK*.

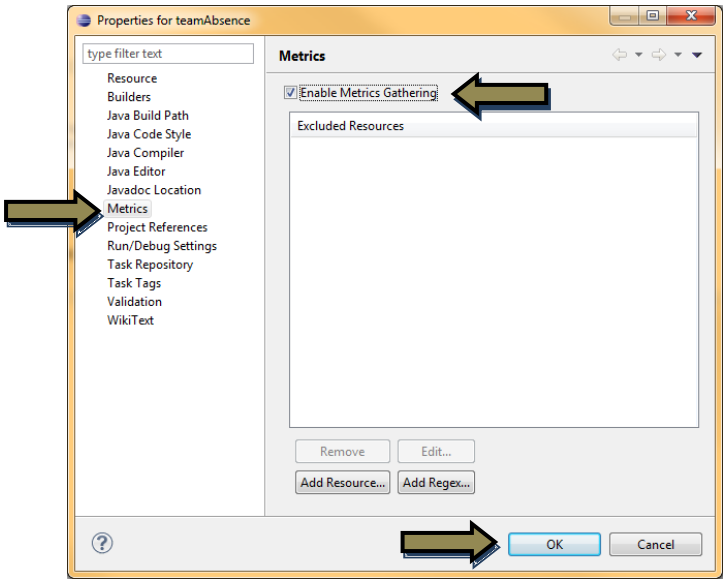


Abbildung 49: Anleitung Eclipse Metrics: Einstellungen 1

Somit sind die Metriken für dieses Projekt aktiviert und Auswertungen können gemacht werden.

4) Um Einstellungen der Grenzwerte vorzunehmen öffnen Sie die Einstellungen von Eclipse unter *Window > Preferences*. Unter dem Punkt *Metrics* können Sie die entsprechenden Einstellungen vornehmen.

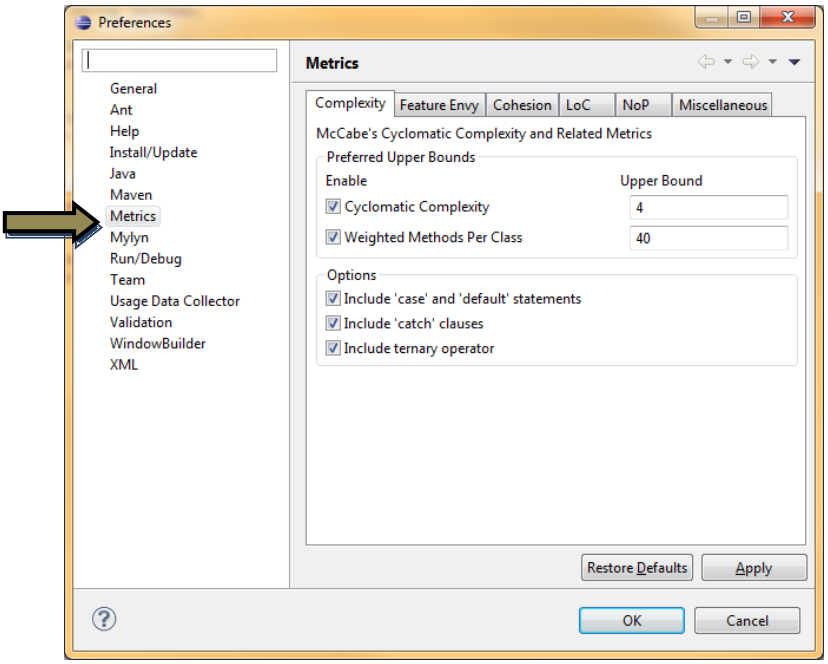


Abbildung 50: Anleitung Eclipse Metrics: Einstellungen 2

7.1.3 Anwendungstipps

7.1.3.1 Programmierung

Während der Programmierung werden Probleme mit Metriken direkt in der Klasse und in der *Problem View* angezeigt.

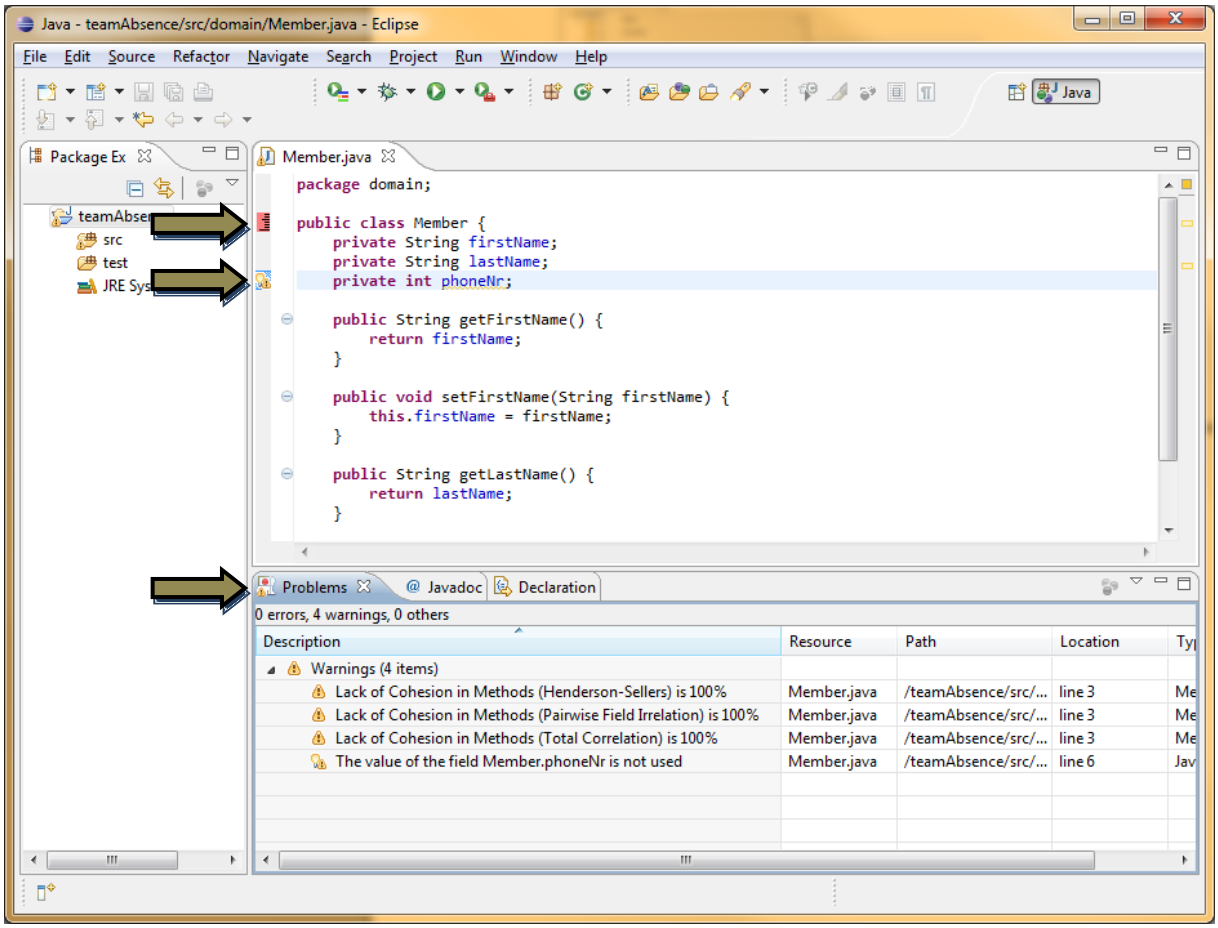


Abbildung 51: Anleitung Eclipse Metrics: Problem View

7.1.3.2 Auswertung

Um eine Auswertung des Projektes zu generieren gehen Sie wie folgt vor:

1) Wählen Sie links das gewünschte Projekt aus und klicken Sie auf *Export…*.

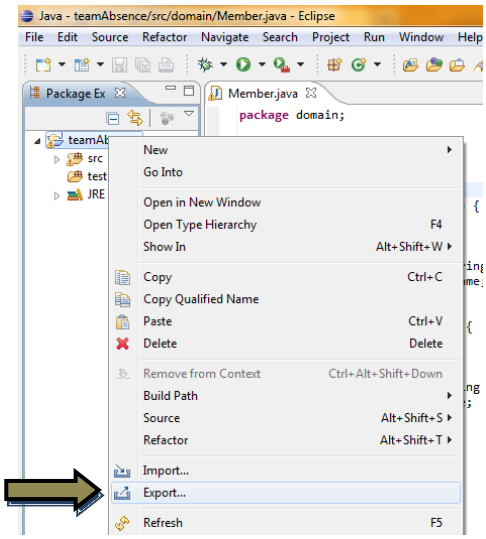


Abbildung 52: Anleitung Eclipse Metrics: Export 1

2) Wählen Sie unter *Other* den Punkt *Metrics* aus und klicken Sie auf *Next*.

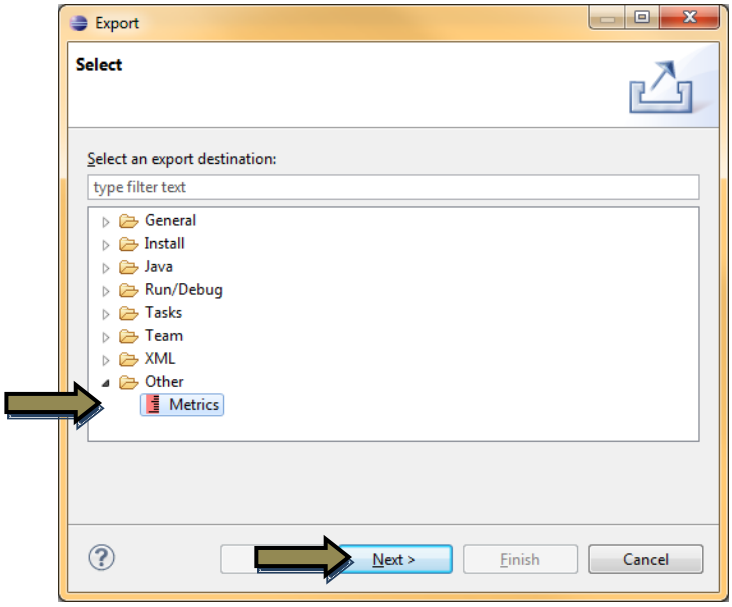


Abbildung 53: Anleitung Eclipse Metrics: Export 2

3) Wählen Sie ein Directory für den Export und Ihre gewünschten Einstellungen und klicken Sie auf *Finish*.

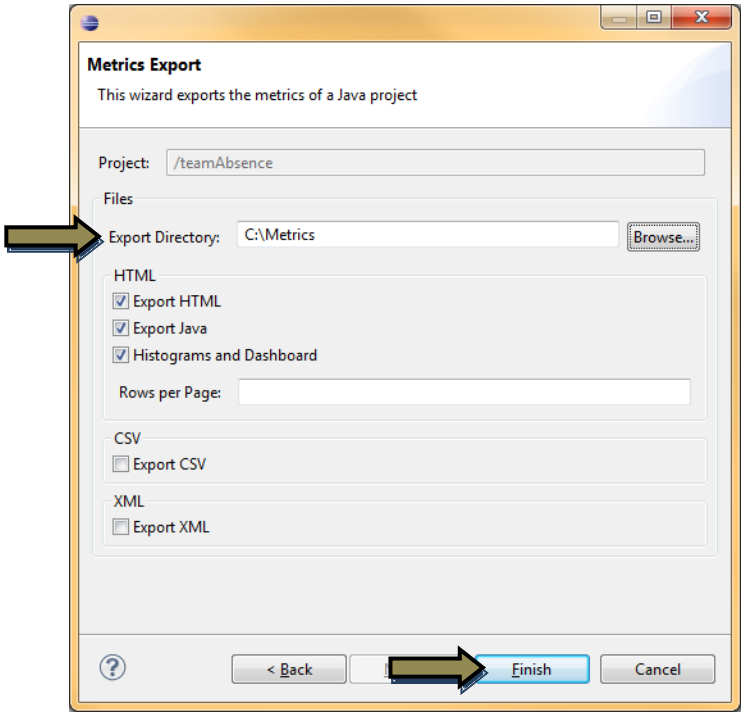


Abbildung 54: Anleitung Eclipse Metrics: Export 3

4) Im gewählten Directory können Sie die *index.html* öffnen und gelangen so zu allen generierten Auswertungen.

7.2 FindBugs

7.2.1 Installation

1) Starten Sie Eclipse.

2) *Eclipse Marketplace* starten.

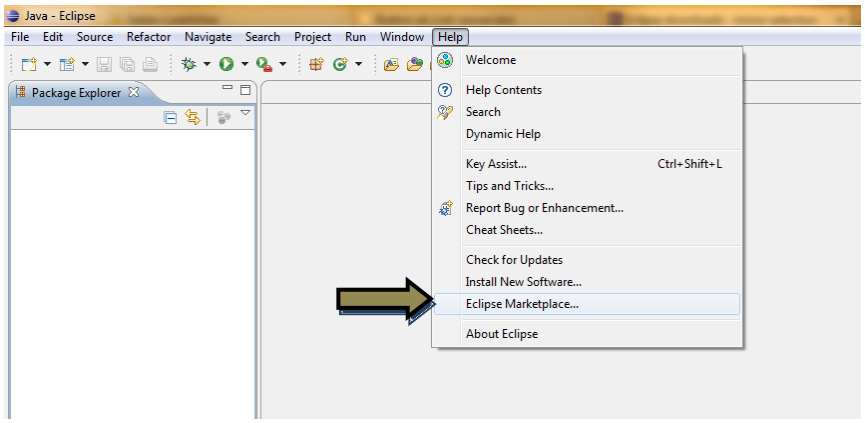


Abbildung 55: Anleitung FindBugs: Eclipse Hilfemenü

3) Suche nach „FindBugs“ und anschliessend auf *Install* klicken.

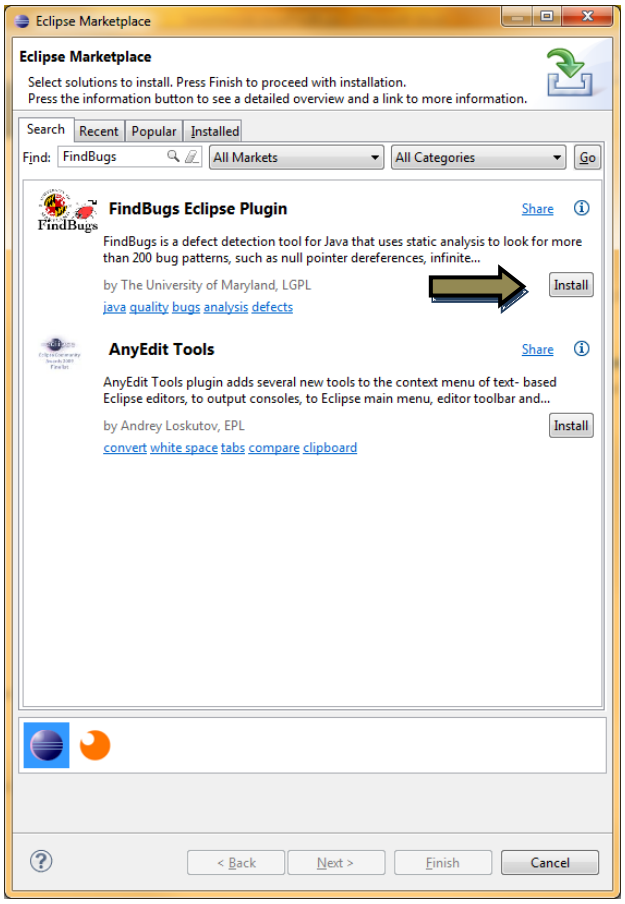


Abbildung 56: Anleitung FindBugs: Eclipse Marketplace

Die entsprechenden Pakete werden geladen, dies kann einen Moment dauern.

4) Klicken Sie auf *Next* und stellen Sie sicher dass alle Elemente ausgewählt sind.

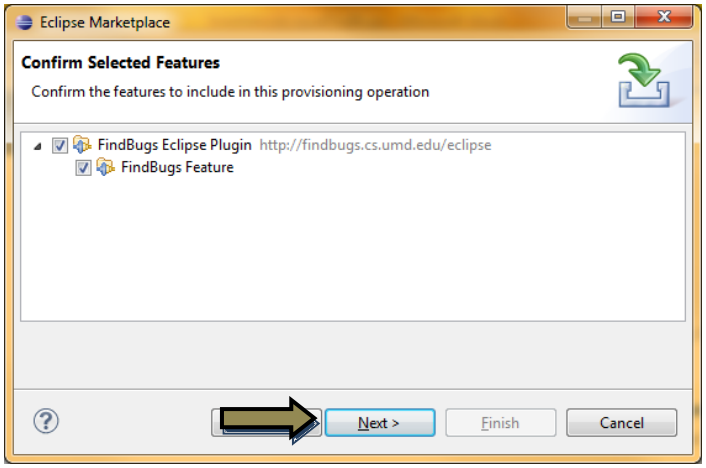


Abbildung 57: Anleitung FindBugs: Download

5) Akzeptieren Sie die Lizenzvereinbarung und klicken Sie auf *Finish*.

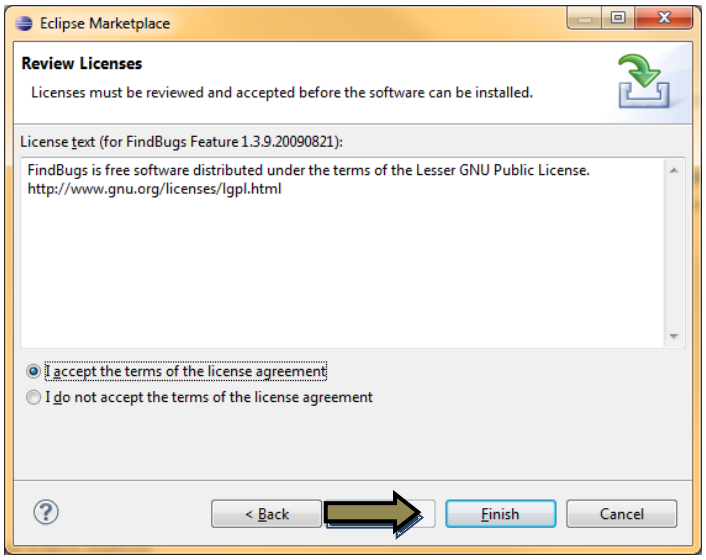


Abbildung 58: Anleitung FindBugs: Lizenzvereinbarung

6) Das Plug-In wird installiert und kann nach einem Neustart von Eclipse verwendet werden.

Es kann sein das während der Installation eine Warnung auftaucht, diese kann jedoch ignoriert werden. Klicken sie auf *OK* und die Installation wird fortgesetzt.

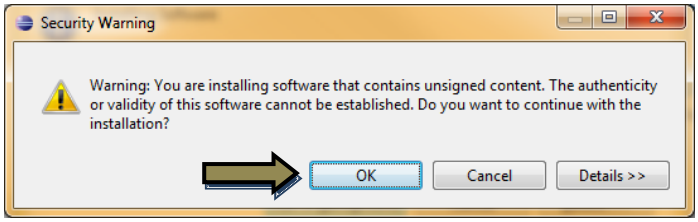


Abbildung 59: Anleitung FindBugs: Unsigned Content Warnung

7.2.2 Setup

1) Starten Sie Eclipse.

2) Wählen Sie links das gewünschte Projekt aus und klicken Sie auf *Properties*. Wählen Sie den Punkt FindBugs, setzen Sie die beiden Haken und klicken Sie auf OK.

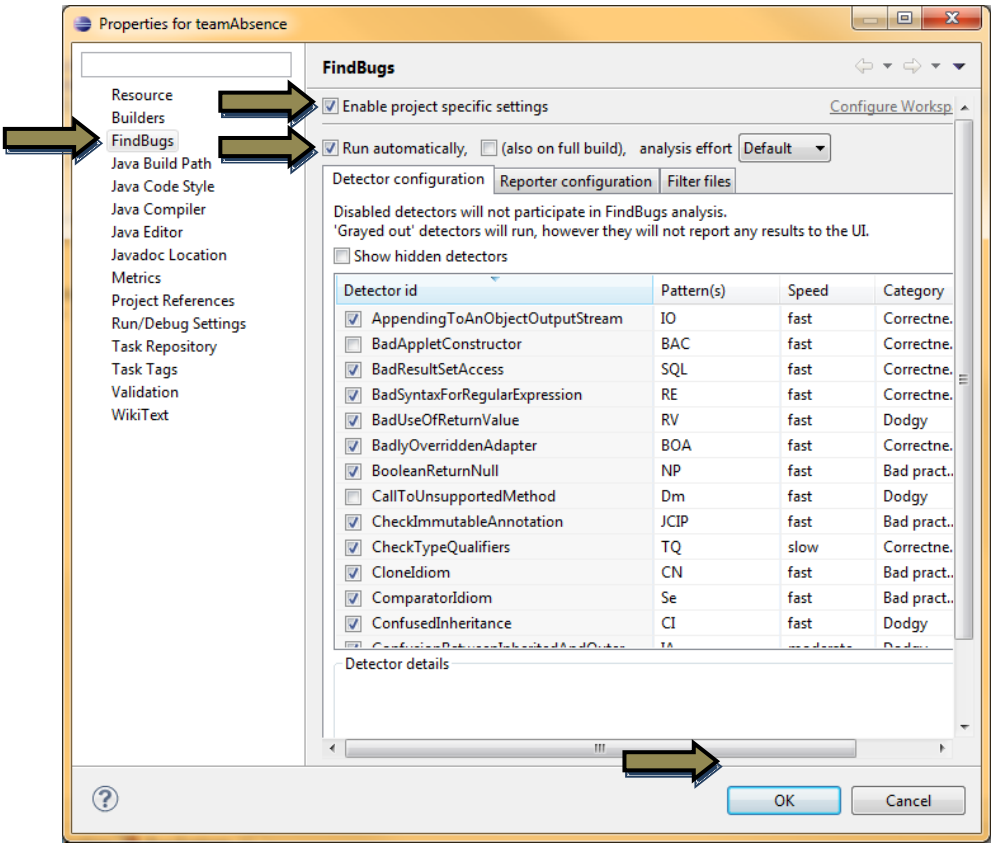


Abbildung 60: Anleitung FindBugs: Settings

Nun ist FindBugs für das entsprechende Projekt aktiviert und beim Speichern einer Datei wird die Analyse durchgeführt.

7.2.3 Anwendungstipps

Während der Programmierung werden Mängel von FindBugs direkt in der entsprechenden Klasse und in der *Problem View* angezeigt.

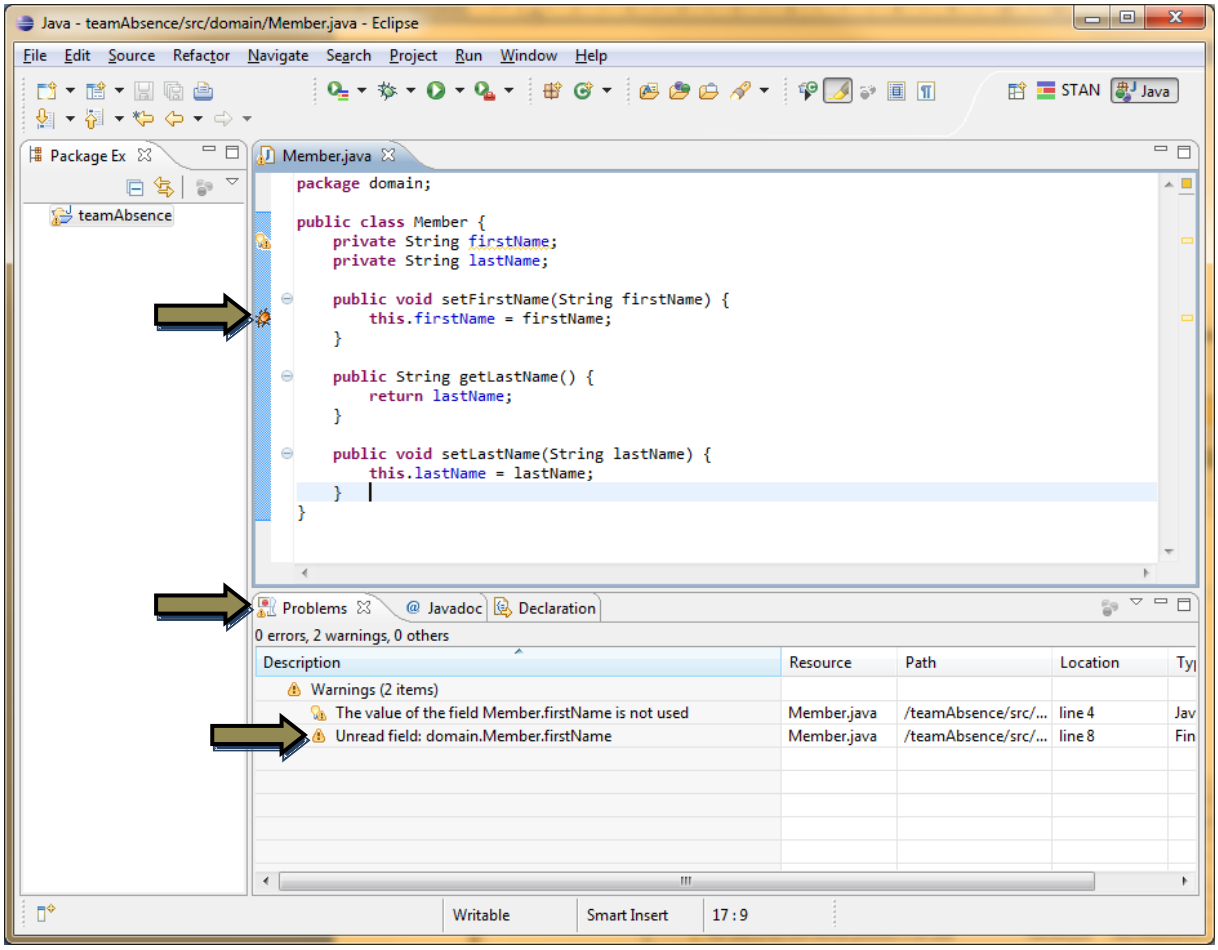


Abbildung 61: Anleitung FindBugs: Problem View

7.3 STAN

7.3.1 Installation

1) Starten Sie Eclipse.

2) *Install New Software…* starten.

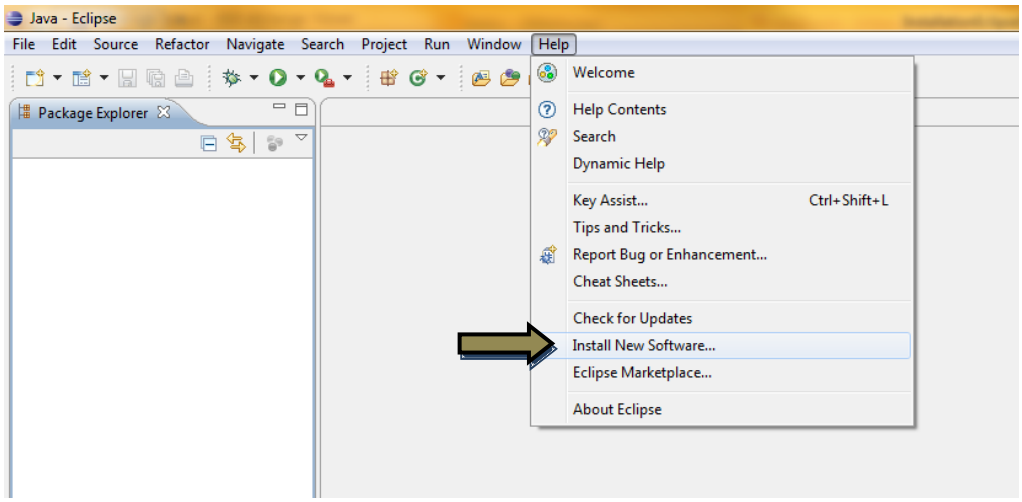


Abbildung 62: Anleitung STAN: Eclipse Help-menü

3) Geben Sie folgenden Link ein: http://update.stan4j.com/ide, wählen Sie alle Pakete aus und klicken Sie auf *Next*.

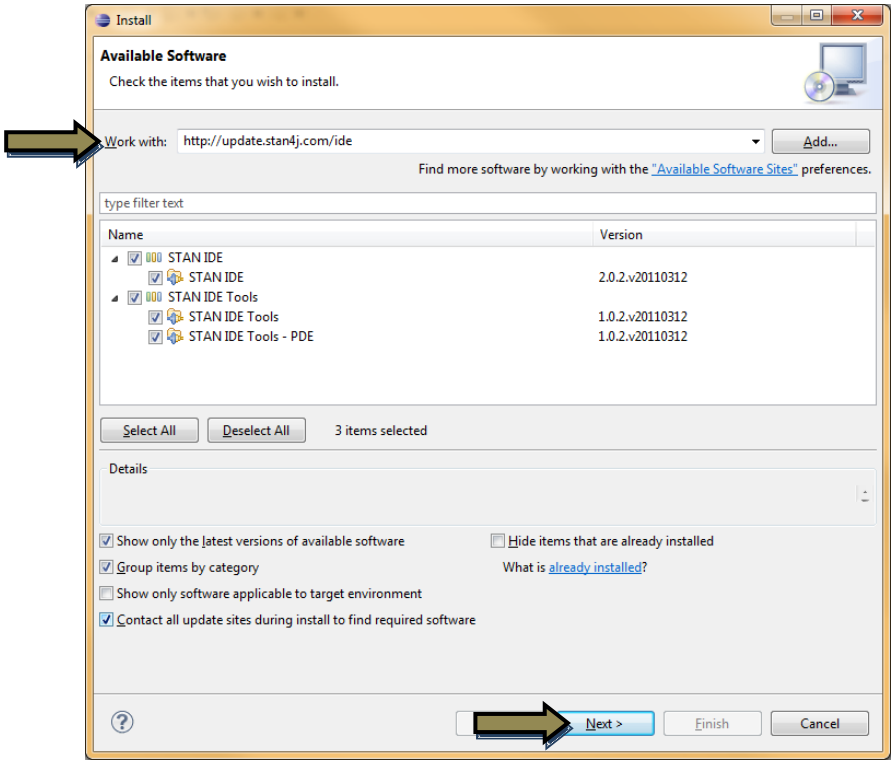


Abbildung 63: Anleitung STAN: Download

Falls Sie mit dem angegebenen Link keine Pakete finden, suchen Sie die aktuelle Downloadseite unter http://stan4j.com/.

4) Klicken Sie nochmals auf *Next*.

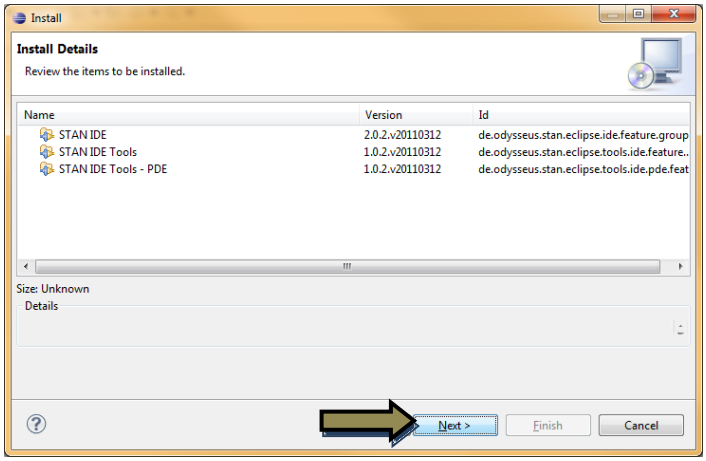


Abbildung 64: Anleitung STAN: Installation

5) Akzeptieren Sie die Lizenzvereinbarung und klicken Sie auf *Finish*.

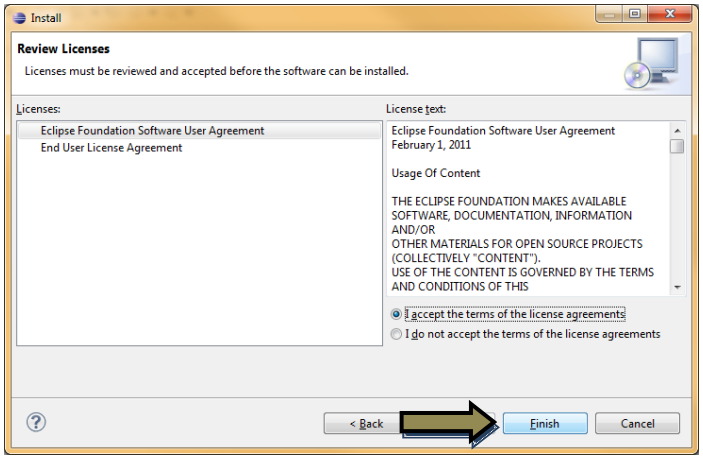


Abbildung 65: Anleitung STAN: Lizenzvereinbarung

6) Das Plug-In wird installiert und kann nach einem Neustart von Eclipse verwendet werden.

Es kann sein das während der Installation eine Warnung auftaucht, diese kann jedoch ignoriert werden. Klicken sie auf *OK* und die Installation wird fortgesetzt.

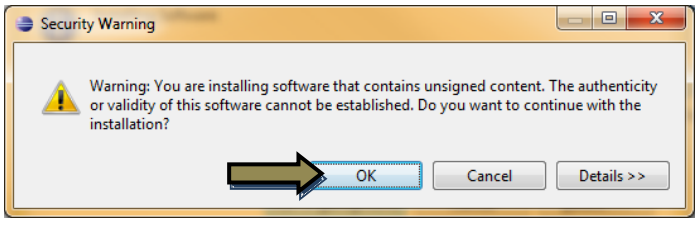


Abbildung 66: Anleitung STAN: Unsigned Content Warnung

7.3.2 Setup

1) Starten Sie Eclipse.

2) Wählen Sie unter *Window > Open Perspective > Other* die Perspektive *STAN* aus und klicken Sie auf *OK*.

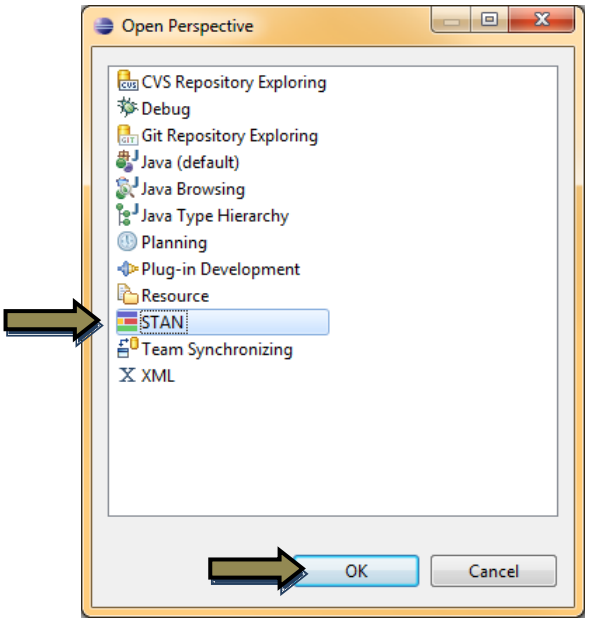


Abbildung 67: Anleitung STAN: Eclipse Perspektiven

Wenn Sie anschliessen Auswertungen ausführen finden Sie die Resultate in der STAN Perspektive.

7.3.3 Anwendungstipps

Während der Anwendung werden keine direkten Auswertungen gemacht. Diese müssen wie folgt ausgeführt werden.

1) Wählen Sie links das gewünschte Projekt aus und klicken Sie auf *Run As > 3 Structure Analysis*.

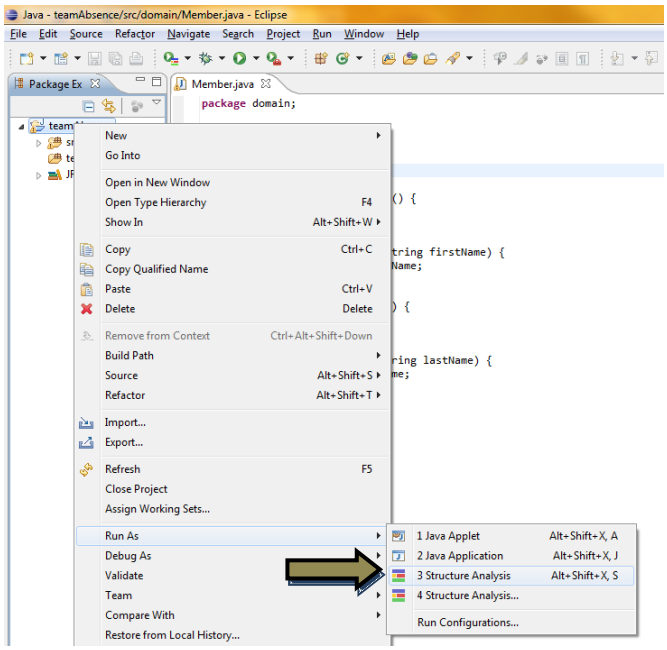


Abbildung 68: Anleitung STAN: Projekt Kontextmenü

Die Auswertungen zum Projekt finden Sie anschliessend in der STAN Perspektive.

2) Um einzelne Punkte der Auswertungen als View zu sehen können Sie die entsprechenden Views unter *Window > Show View > Other* und dort unter dem Punkt *STAN* finden.

7.4 WindowBuilder Pro

7.4.1 Installation

1) Starten Sie Eclipse.

2) *Install New Software…* starten.

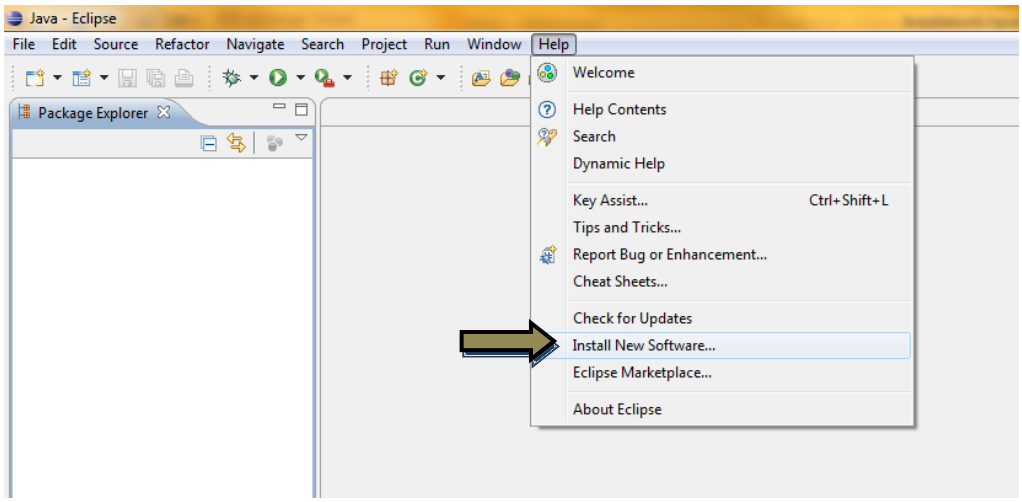


Abbildung 69: Anleitung WindowBuilder Pro: Eclipse Install New Software

3) Holen Sie sich den Update Link für Ihre entsprechende Eclipse Version unter folgender URL: http://code.google.com/intl/de-DE/eclipse/docs/download.html, wählen Sie das Paket ‚GWT Designer for GPE' aus und klicken Sie auf *Next*.

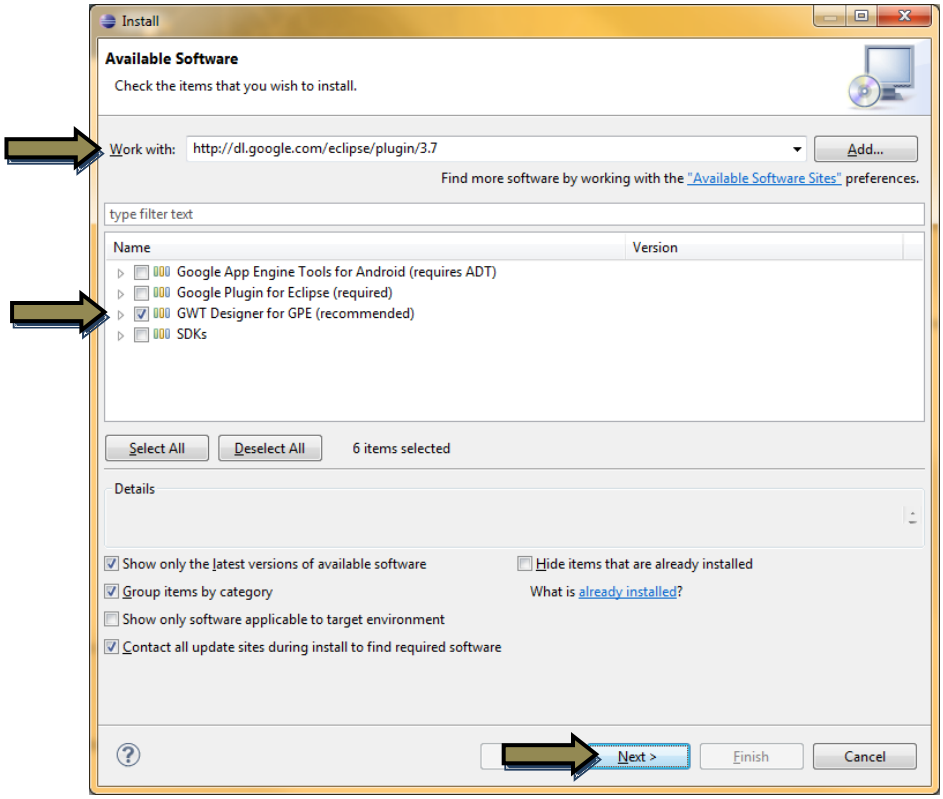


Abbildung 70: Anleitung WindowBuilder Pro: Download

4) Klicken Sie nochmals auf *Next*.

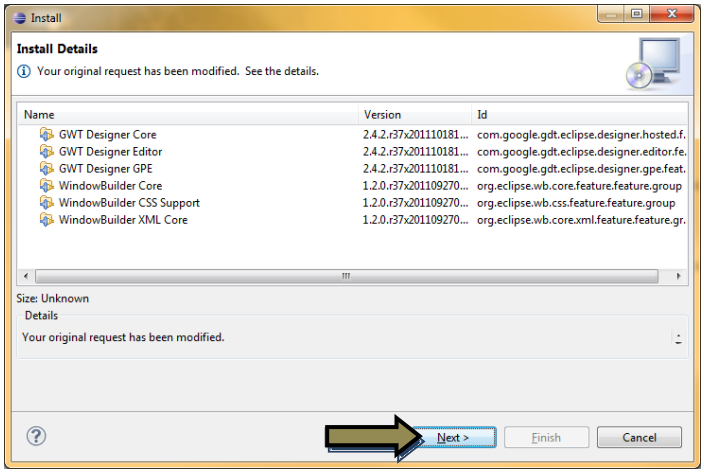


Abbildung 71: Anleitung WindowBuilder Pro: Installation

5) Akzeptieren Sie die Lizenzvereinbarung und klicken Sie auf *Finish*.

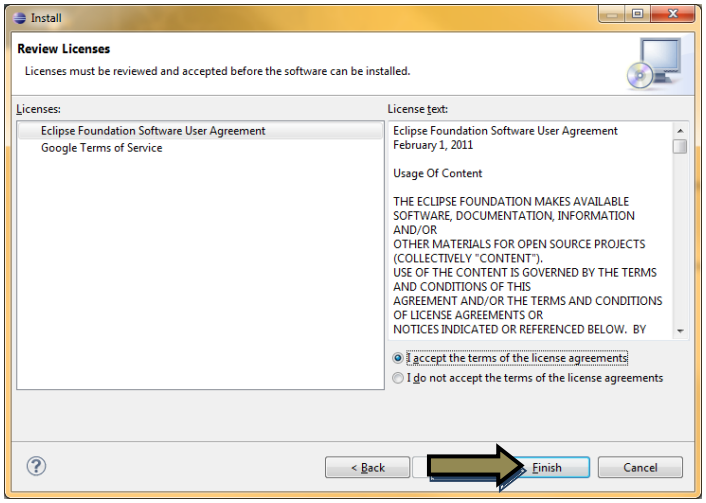


Abbildung 72: Anleitung WindowBuilder Pro: Lizenzvereinbarung

6) Das Plug-In wird installiert und kann nach einem Neustart von Eclipse verwendet werden.

Es kann sein das während der Installation eine Warnung auftaucht, diese kann jedoch ignoriert werden. Klicken sie auf *OK* und die Installation wird fortgesetzt.

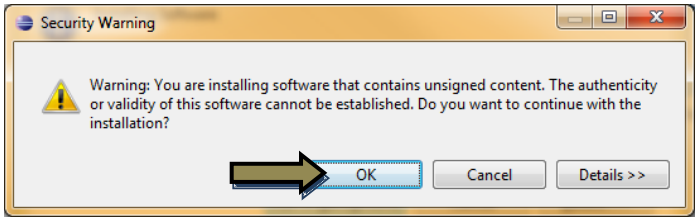


Abbildung 73: Anleitung WindowBuilder Pro: Unsigned Content Warnung

7.4.2 Setup

Es ist kein Setup nötig, WindowBuilder Pro kann direkt verwendet werden.

7.4.3 Anwendungstipps

Um eine View erstellen zu können klicken Sie auf dem entsprechenden Package auf *New > Other…* und Sie gelangen zu folgender Ansicht. Unter *WindowBuilder* finden Sie alle erstellbaren Views. Wählen Sie sich die entsprechende aus und machen Sie mit dem Wizard weiter.

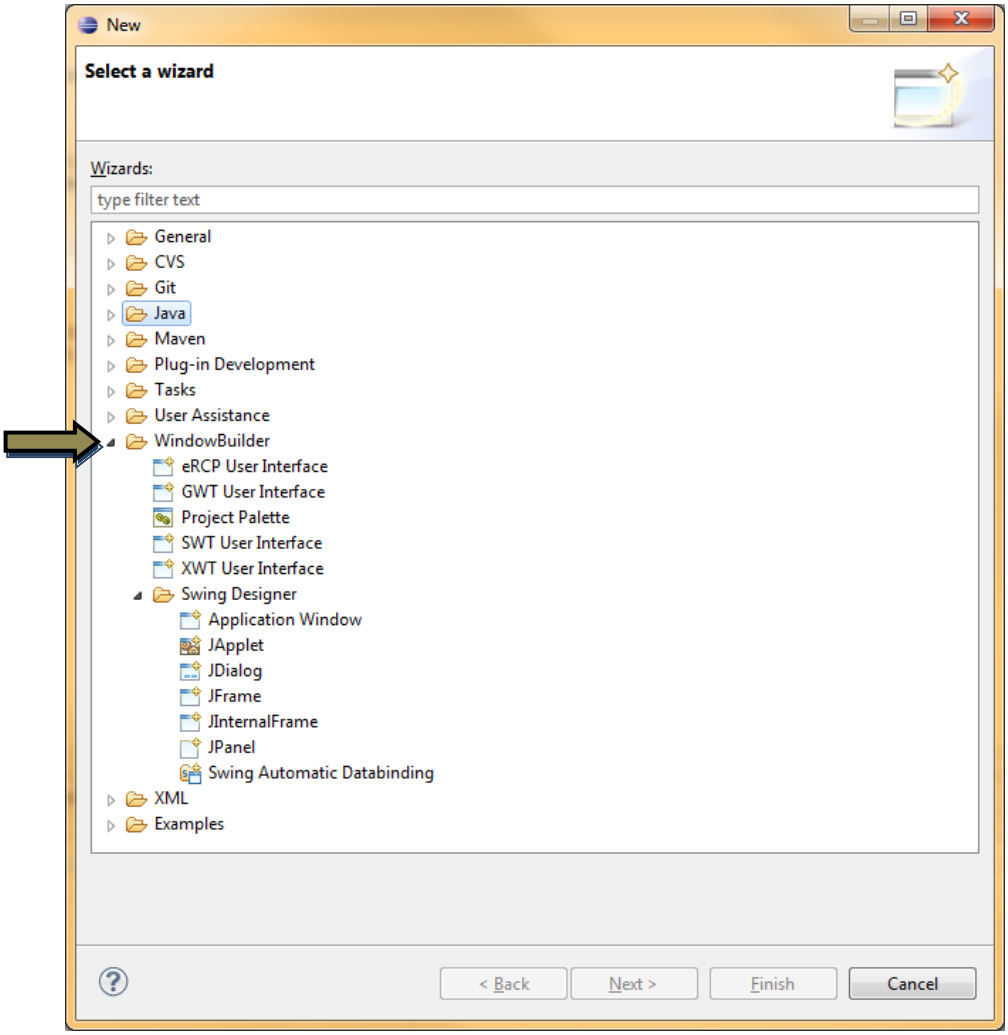


Abbildung 74: Anleitung WindowBuilder Pro: Eclipse New Wizard

7.5 Checkstyle

7.5.1 Installation

1) Starten Sie Eclipse.

2) *Eclipse Marketplace* starten.

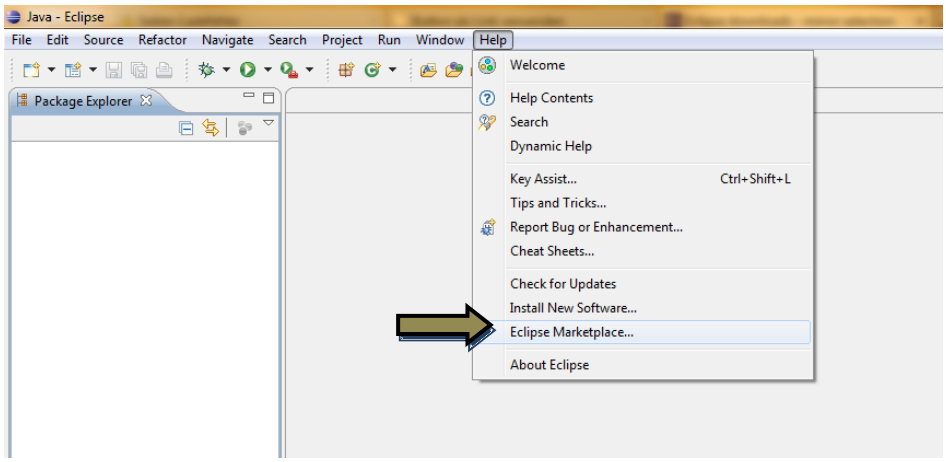


Abbildung 75: Anleitung Checkstyle: Eclipse Hilfemenü

3) Suche nach „Checkstyle“ und anschliessend auf *Install* klicken.

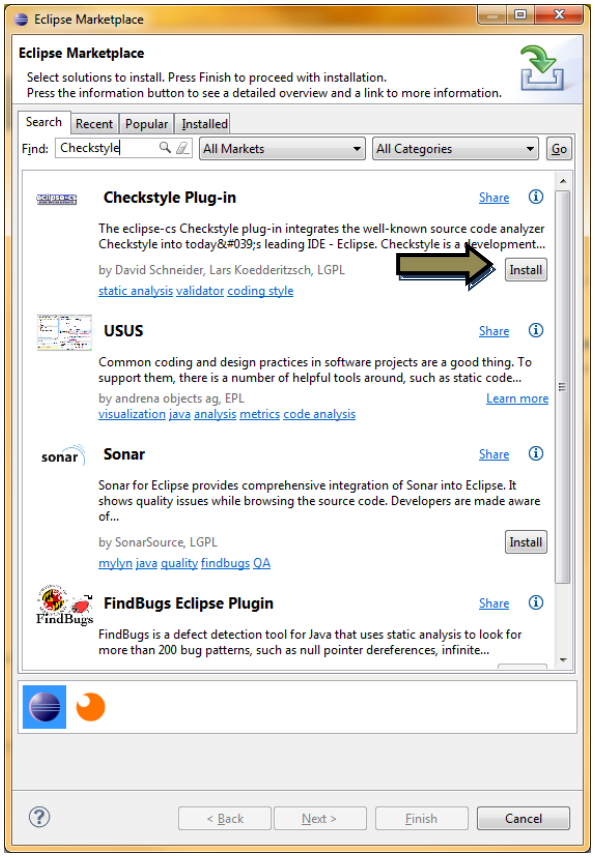


Abbildung 76: Anleitung Checkstyle: Eclipse Marketplace

Die entsprechenden Pakete werden geladen, dies kann einen Moment dauern.

4) Klicken Sie auf *Next* und stellen Sie sicher dass alle Elemente ausgewählt sind.

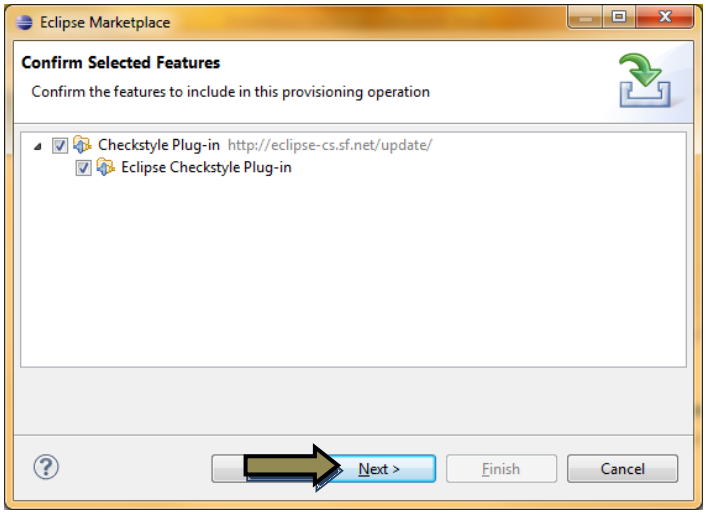


Abbildung 77: Anleitung Checkstyle: Download

5) Akzeptieren Sie die Lizenzvereinbarung und klicken Sie auf *Finish*.

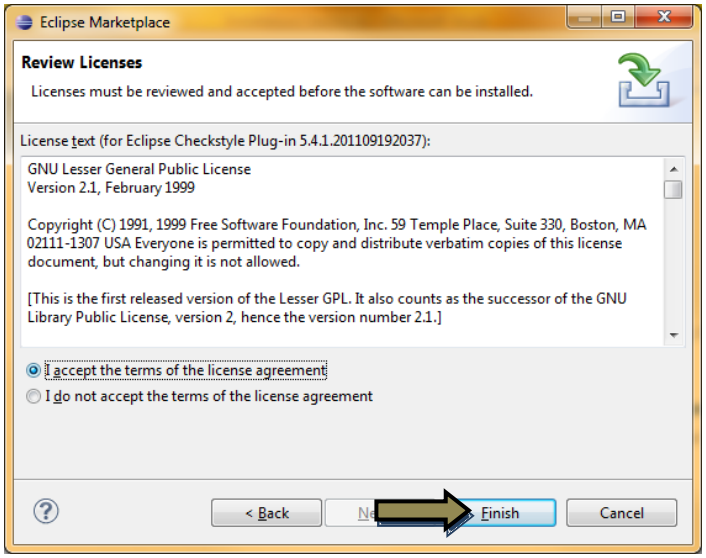


Abbildung 78: Anleitung Checkstyle: Lizenzvereinbarung

6) Das Plug-In wird installiert und kann nach einem Neustart von Eclipse verwendet werden.

Es kann sein das während der Installation eine Warnung auftaucht, diese kann jedoch ignoriert werden. Klicken sie auf *OK* und die Installation wird fortgesetzt.

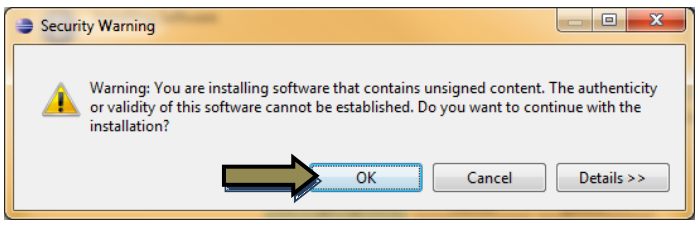


Abbildung 79: Anleitung Checkstyle: Unsigned Content Warnung

7.5.2 Setup

1) Starten Sie Eclipse.

2) Wählen Sie links das gewünschte Projekt aus und klicken Sie auf *Properties*.

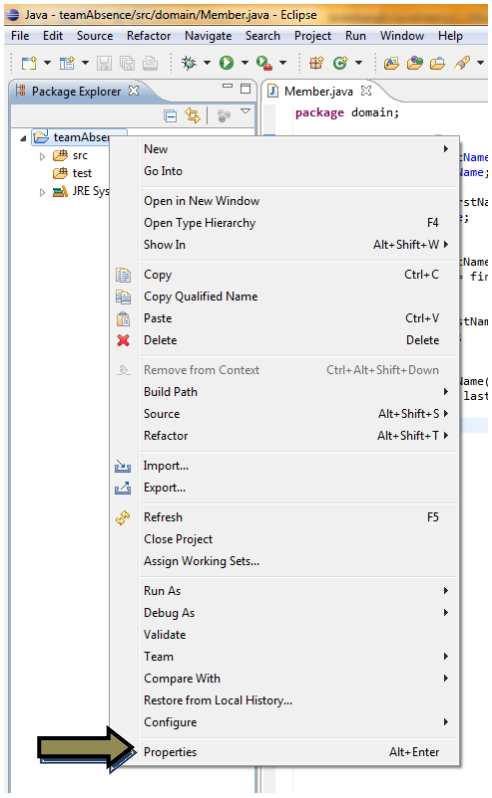


Abbildung 80: Anleitung Checkstyle: Projektkontextmenü

3) Wählen Sie den Punkt *Checkstyle*. Anschliessend setzen Sie einen Haken bei *Checkstyle active for this project* und klicken Sie auf *OK*.

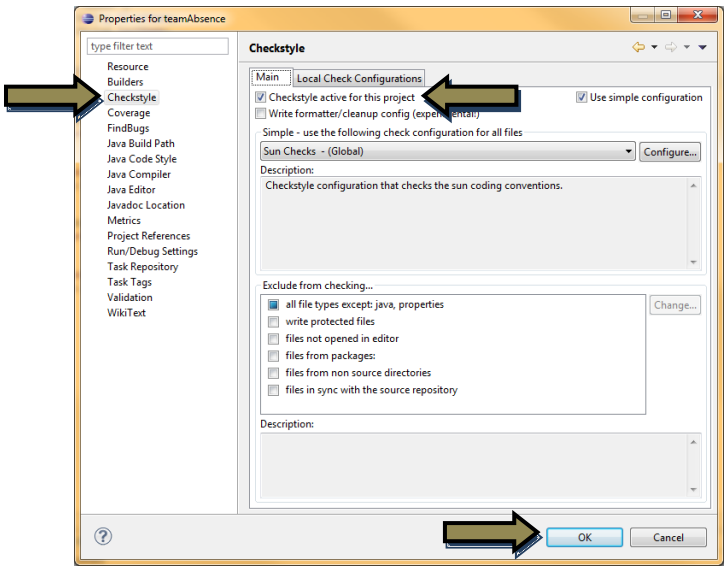


Abbildung 81: Anleitung Checkstyle: Einstellungen 1

Somit ist Checkstyle für dieses Projekt aktiviert und Auswertungen werden gemacht.

4) Um Einstellungen von Checkstyle vorzunehmen öffnen Sie die Einstellungen von Eclipse unter *Window > Preferences*. Unter dem Punkt *Checkstyle* wählen Sie *Sun Checks* (sind aktiviert) und klicken Sie auf *Configure…*.

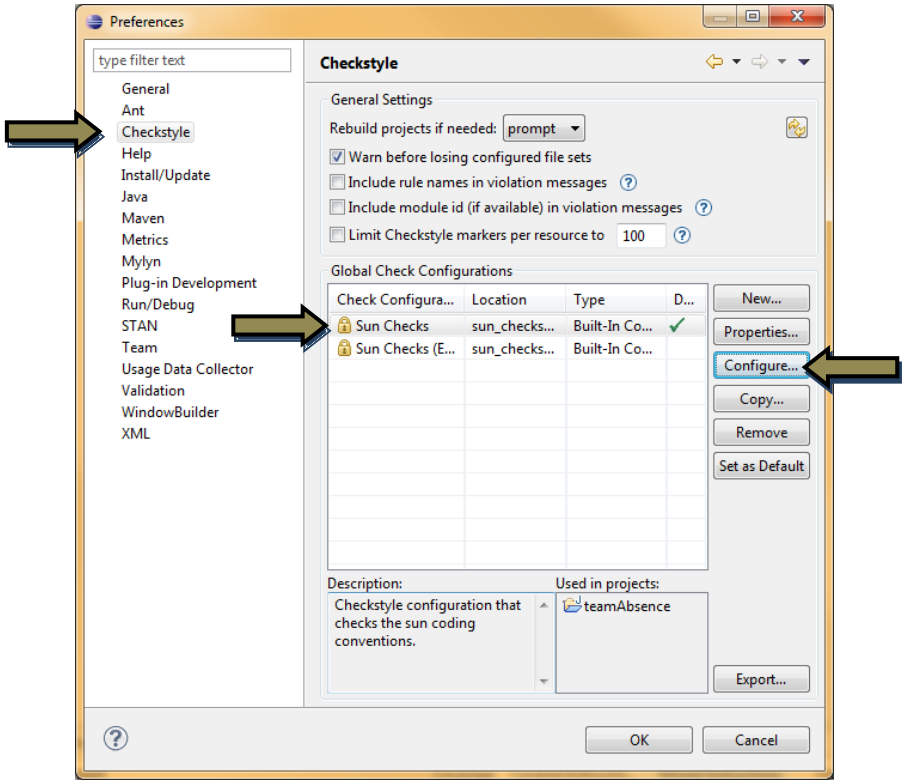


Abbildung 82: Anleitung Checkstyle: Einstellungen 2

7.5.3 Anwendungstipps

Die Probleme die von Checkstyle werden direkt in der Klasse und in der *Problem View* angezeigt.

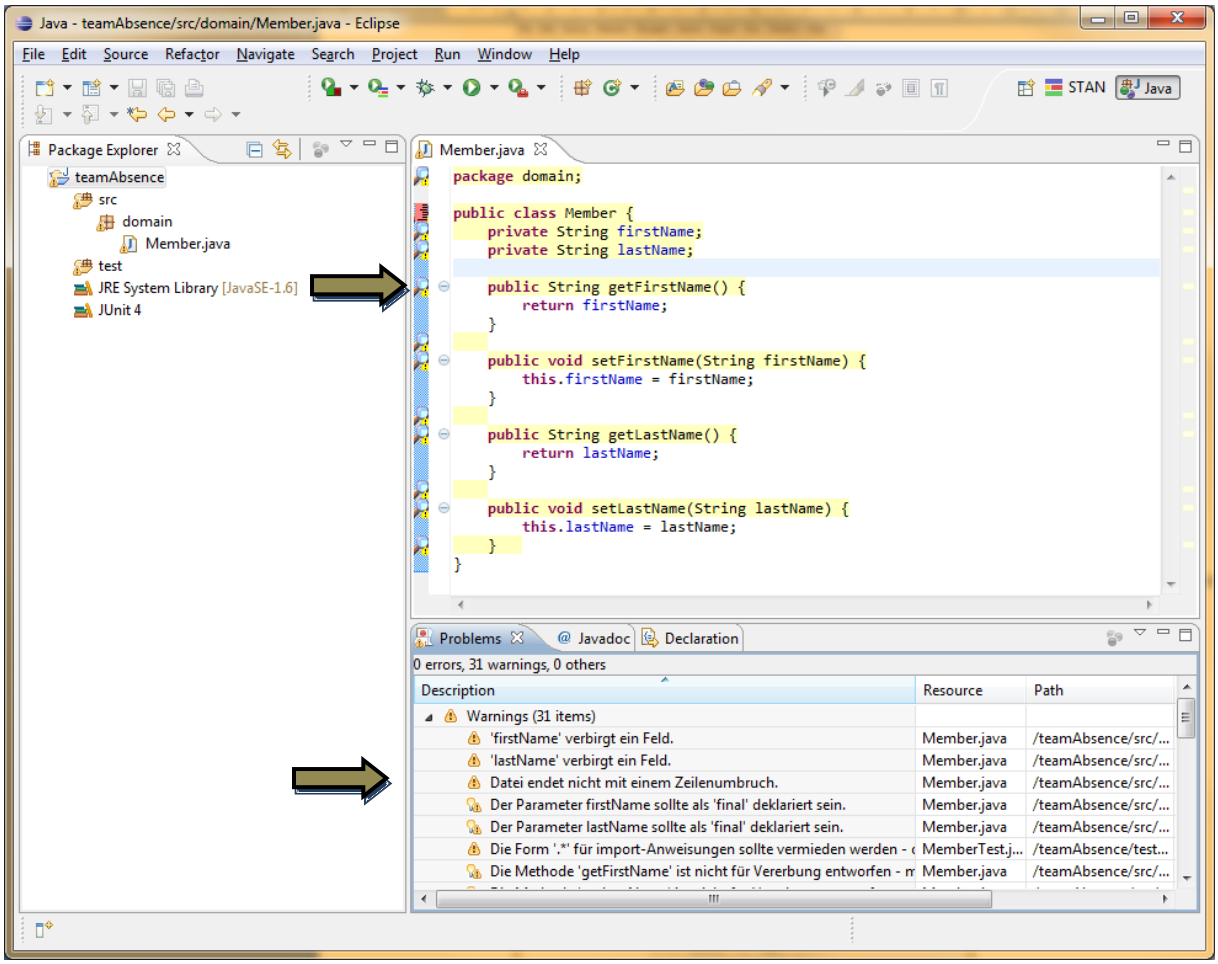


Abbildung 83: Anleitung Checkstyle: Problem View

Wie Sie sehen wird sehr viel als falsch angezeigt. Es wird deshalb empfohlen, entweder die Einstellungen dementsprechend anzupassen (Unter Setup, Punkt 4) oder Checkstyle nicht zu verwenden.

7.6 EclEmma

7.6.1 Installation

1) Starten Sie Eclipse.

2) *Eclipse Marketplace* starten.

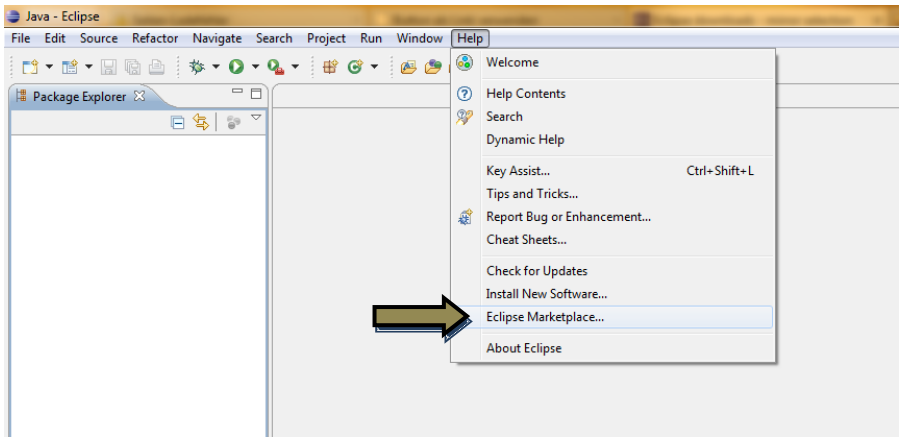


Abbildung 84: Anleitung EclEmma: Eclipse Hilfemenü

3) Suche nach „EclEmma“ und anschliessend auf *Install* klicken.

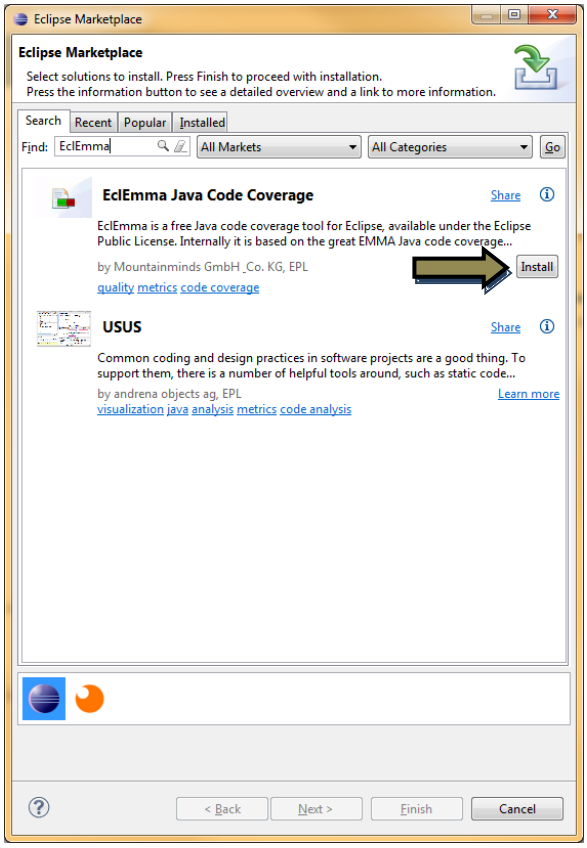


Abbildung 85: Anleitung EclEmma: Eclipse Marketplace

Die entsprechenden Pakete werden geladen, dies kann einen Moment dauern.

4) Klicken Sie auf *Next* und stellen Sie sicher dass alle Elemente ausgewählt sind.

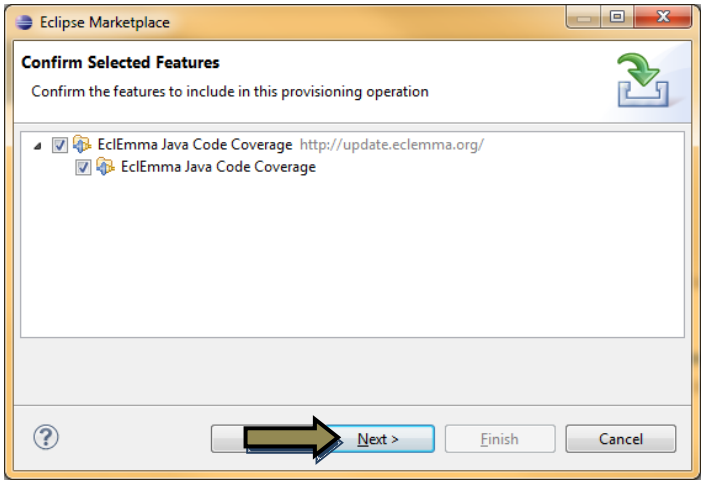


Abbildung 86: Anleitung EclEmma: Download

5) Akzeptieren Sie die Lizenzvereinbarung und klicken Sie auf *Finish*.

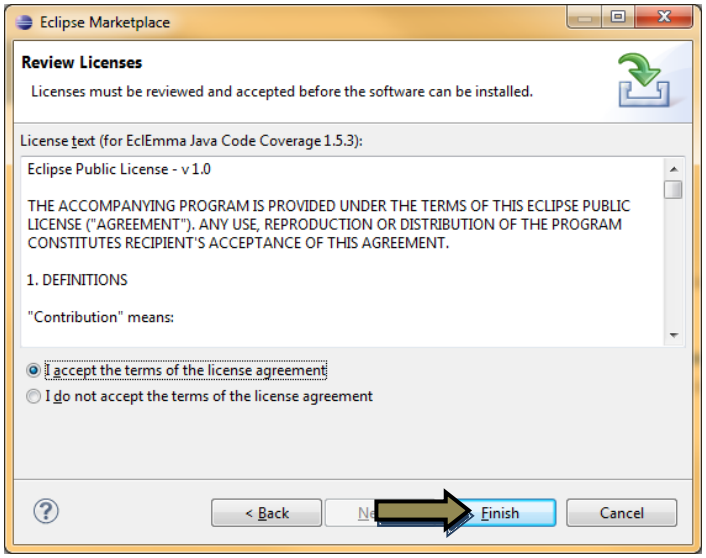


Abbildung 87: Anleitung EclEmma: Lizenzvereinbarung

6) Das Plug-In wird installiert und kann nach einem Neustart von Eclipse verwendet werden.

7.6.2 Setup

Es ist kein Setup nötig, EclEmma kann direkt verwendet werden.

7.6.3 Anwendungstipps

Um die Testabdeckung zu überprüfen und gleichzeitig die Unit Tests auszuführen klicken Sie auf das folgende Icon, wählen Sie im folgenden Fenster *JUnit Test* und klicken Sie auf *OK*.

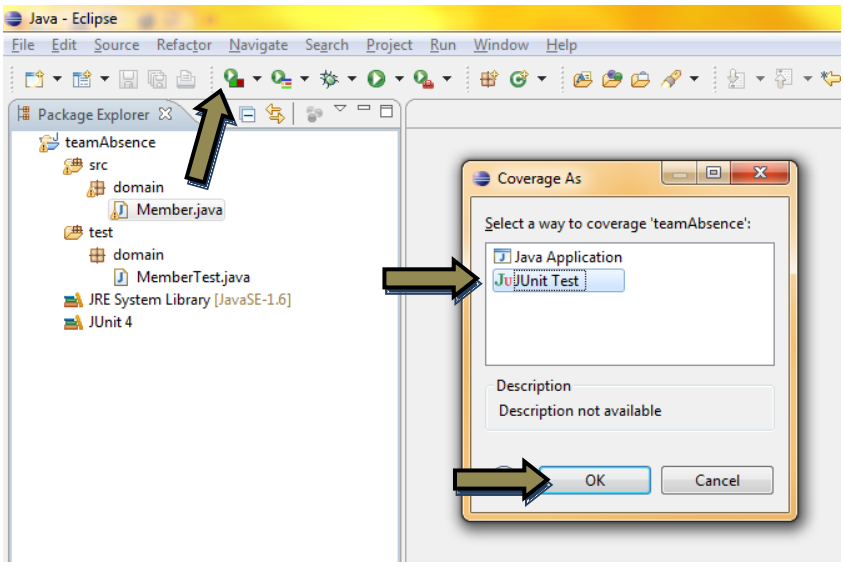


Abbildung 88: Anleitung EclEmma: Run EclEmma

Anschliessend öffnet sich die *Coverage View* und Sie erhalten folgende Auswertungen.

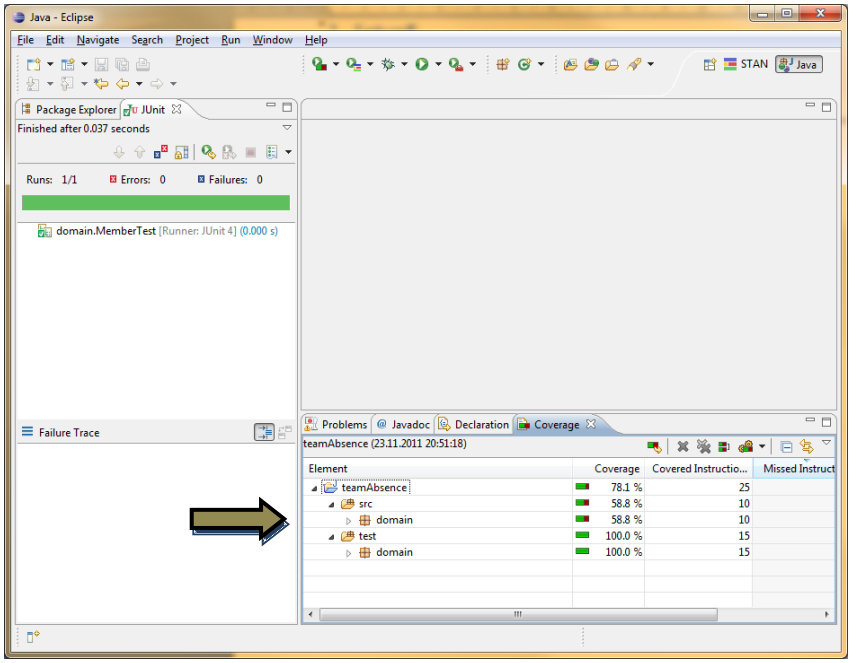


Abbildung 89: Anleitung EclEmma: Coverage View

Nach der Ausführung finden Sie Hinweise auch direkt im Code. Sie sehen direkt, welcher Teil abgedeckt ist mit Tests (grün) und welcher nicht (rot). Diese Markierungen verschwinden allerdings beim Erweitern der Klasse wieder.

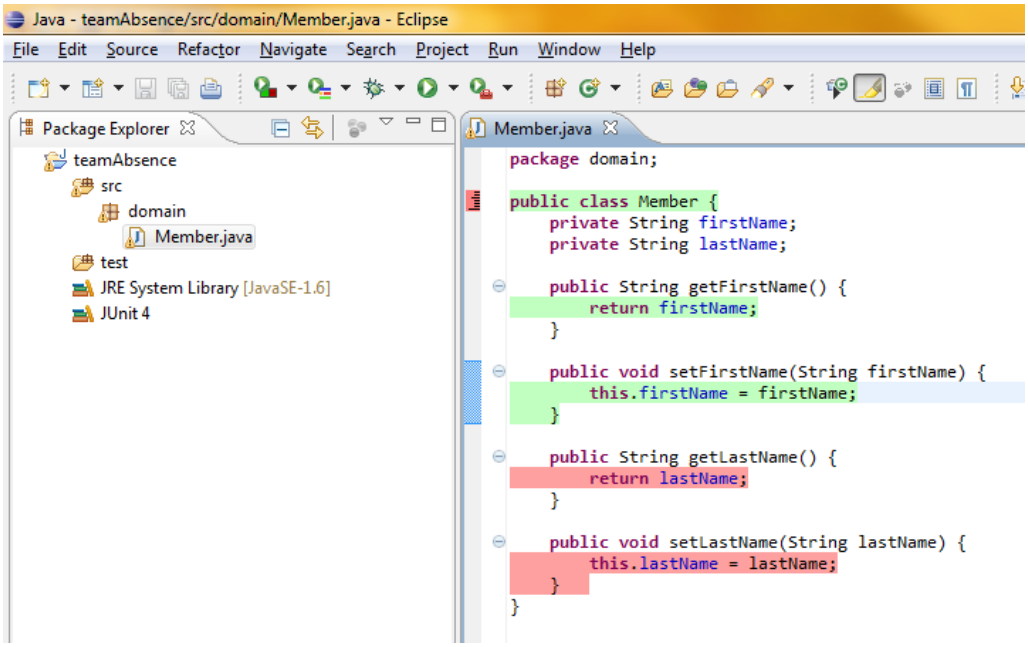


Abbildung 90: Anleitung EclEmma: Testabdeckung im Code

8. SVN (Eclipse Integration)

8.1.1 Installation

1) Starten Sie Eclipse.

2) *Eclipse Marketplace* starten.

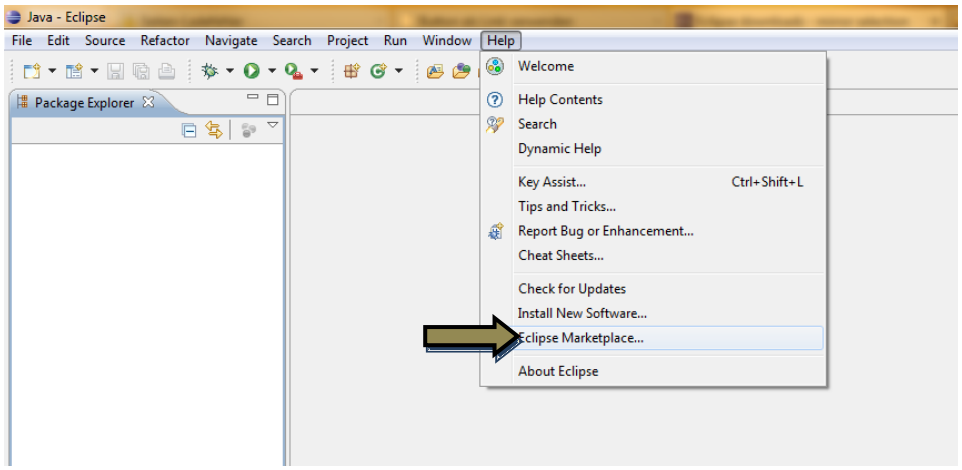


Abbildung 91: Anleitung SVN Eclipse Integration: Eclipse Hilfemenü

3) Suche nach „SVN“ und anschliessend auf *Install* klicken.

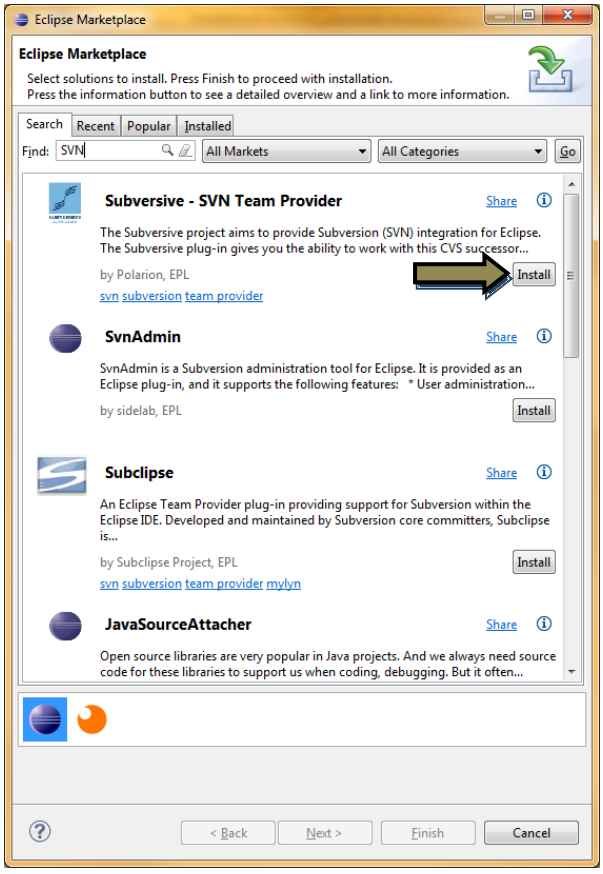


Abbildung 92: Anleitung SVN Eclipse Integration: Eclipse Marketplace SVN-Provider

Die entsprechenden Pakete werden geladen, dies kann einen Moment dauern.

4) Klicken Sie auf *Next* und stellen Sie sicher dass alle Elemente ausgewählt sind.

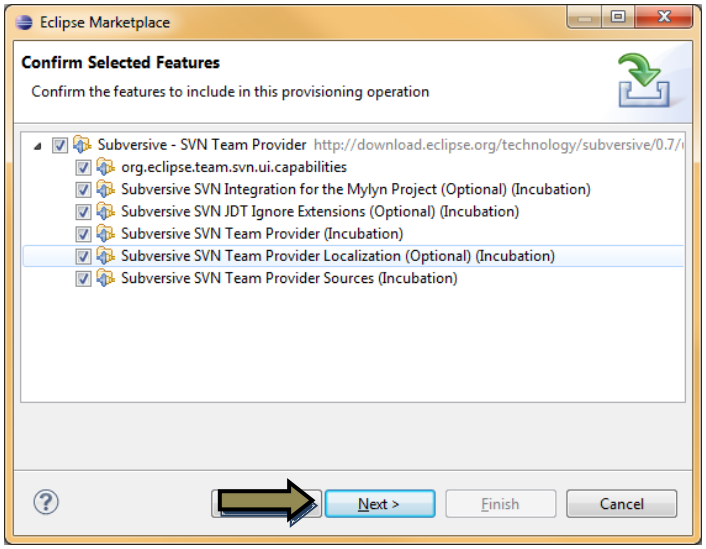


Abbildung 93: Anleitung SVN Eclipse Integration: SVN-Provider Download

5) Akzeptieren Sie die Lizenzvereinbarung und klicken Sie auf *Finish*.

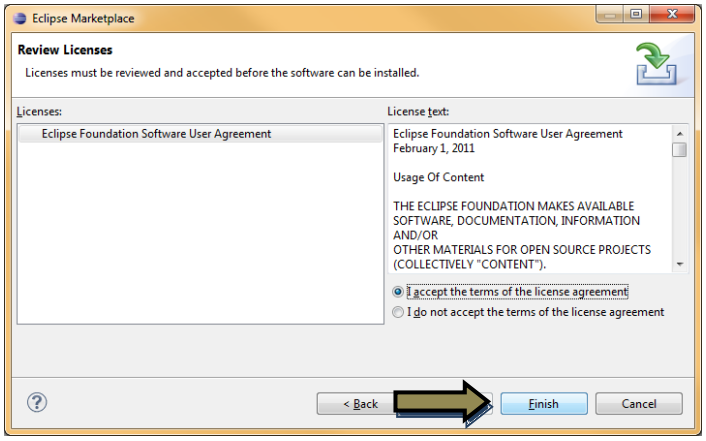


Abbildung 94: Anleitung SVN Eclipse Integration: SVN-Provider Lizenzvereinbarung

6) Das Plug-In wird installiert und nach einem Neustart von Eclipse muss noch ein Connector ausgewählt werden. Wählen Sie einen Connector aus und klicken Sie auf *Finish*.

WICHTIG: Stellen Sie sicher, dass Sie den korrekten Connector für Ihre SVN Version auswählen.

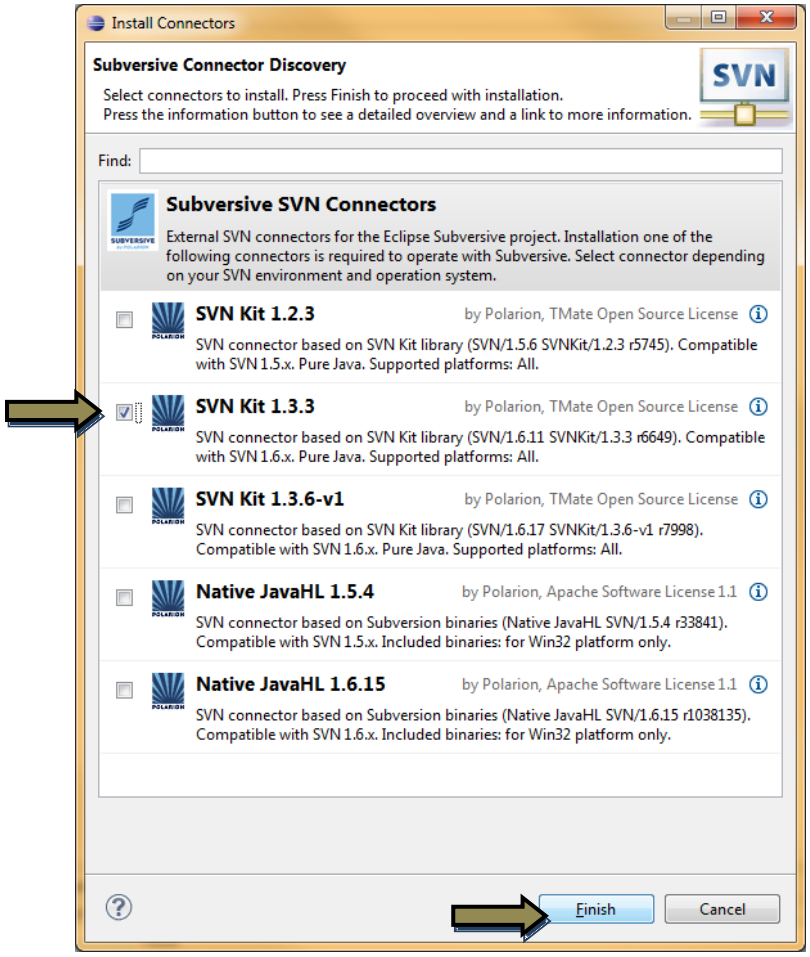


Abbildung 95: Anleitung SVN Eclipse Integration: Eclipse Marketplace SVN-Connector

7) Wählen Sie anschliessend die gewünschten Pakete aus und klicken Sie zweimal auf *Next*.

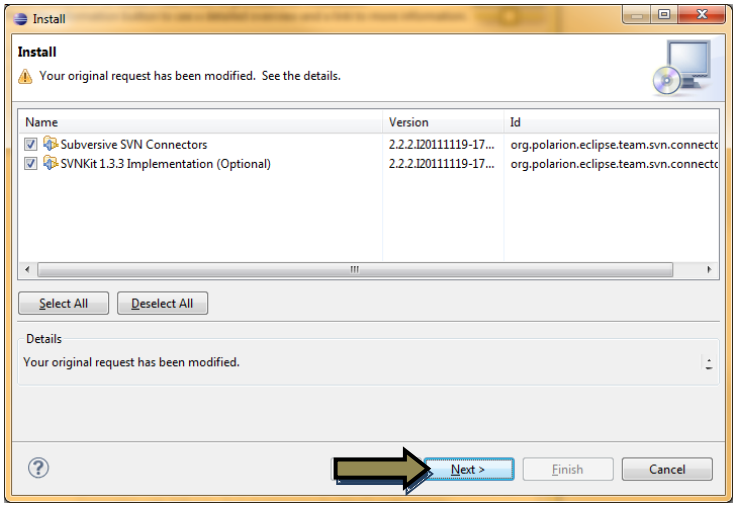


Abbildung 96: Anleitung SVN Eclipse Integration: SVN-Connector Download

8) Akzeptieren Sie die Lizenzvereinbarung und klicken Sie auf *Finish*.

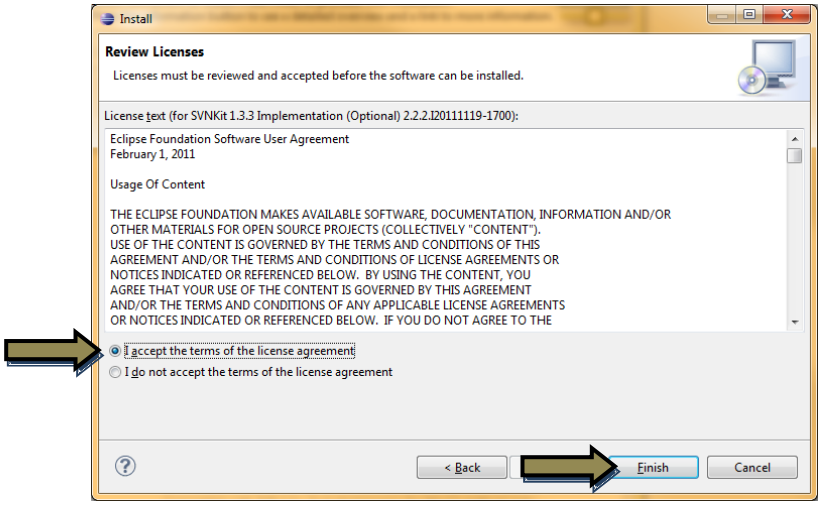


Abbildung 97: Anleitung SVN Eclipse Integration: SVN-Connector Lizenzvereinbarung

9) Das Connector Plug-In wird installiert und kann nach einem erneuten Neustart von Eclipse verwendet werden.

Es kann sein das während der Installation eine Warnung auftaucht, diese kann jedoch ignoriert werden. Klicken sie auf *OK* und die Installation wird fortgesetzt.

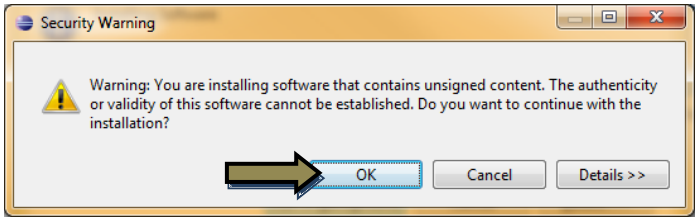


Abbildung 98: Anleitung SVN Eclipse Integration: SVN-Connector Unsigned Content Warnung

8.1.2 Setup

8.1.2.1 Setup Projekt / Projekt teilen

Es geht hier darum ein Projekt zu erstellen und dieses für andere Teammitglieder verfügbar zu machen. Die folgenden Schritte dienen dazu:

1) Erstellen Sie zuerst ein neues Projekt mit den entsprechenden Source Ordnern (src, test).

2) Anschliessend klicken Sie auf dem erstellten Projekt *Team > Share Project…*.

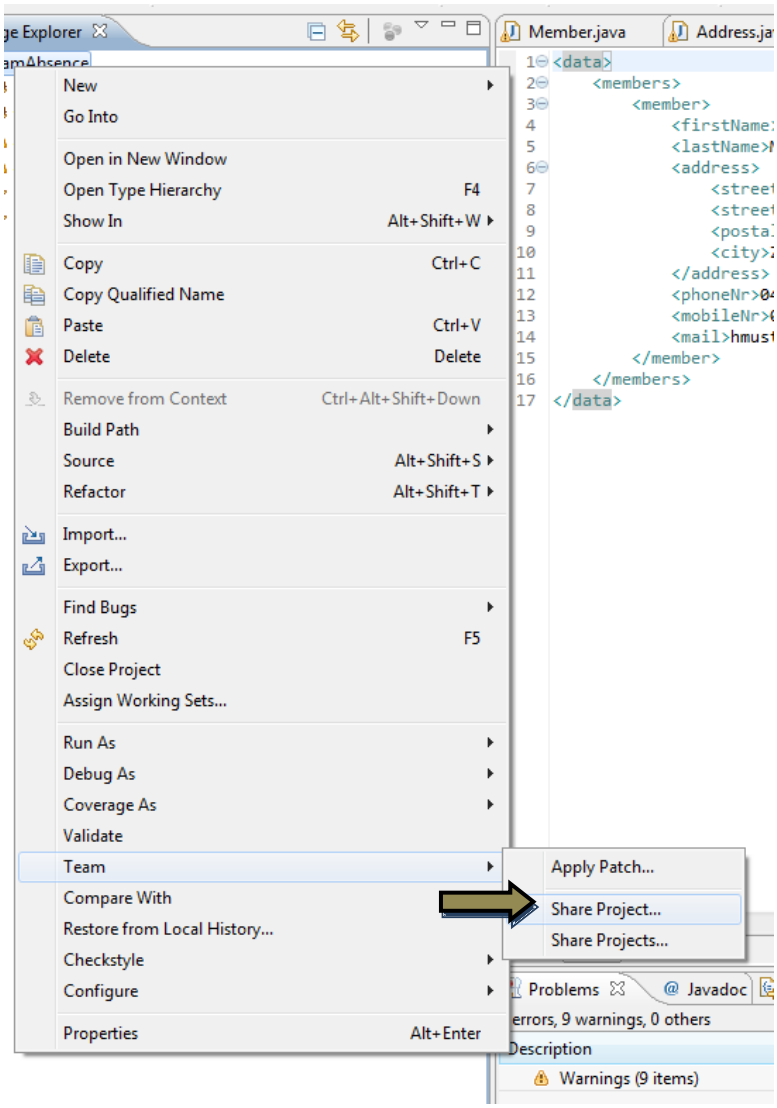


Abbildung 99: Anleitung SVN Eclipse Integration: Projektkontextmenü

3) Wählen Sie im folgenden Fenster *SVN* aus und klicken Sie auf *Next*.

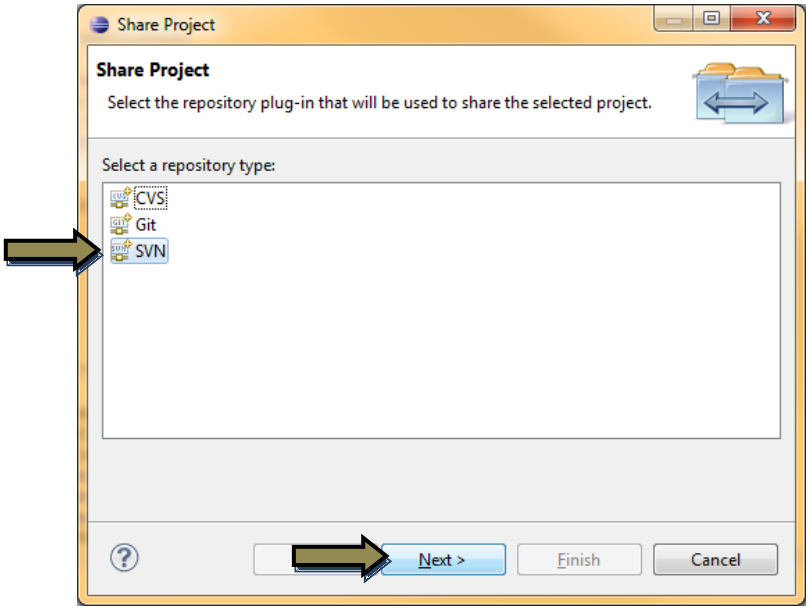


Abbildung 100: Anleitung SVN Eclipse Integration: Share Project

4) Geben Sie die URL folgendermassen ein:
svn://svns.hsr.ch/<repository-name>/<source-ordner>

Ausserdem den Benutzer und das Passwort und klicken Sie anschliessend auf *Next*.

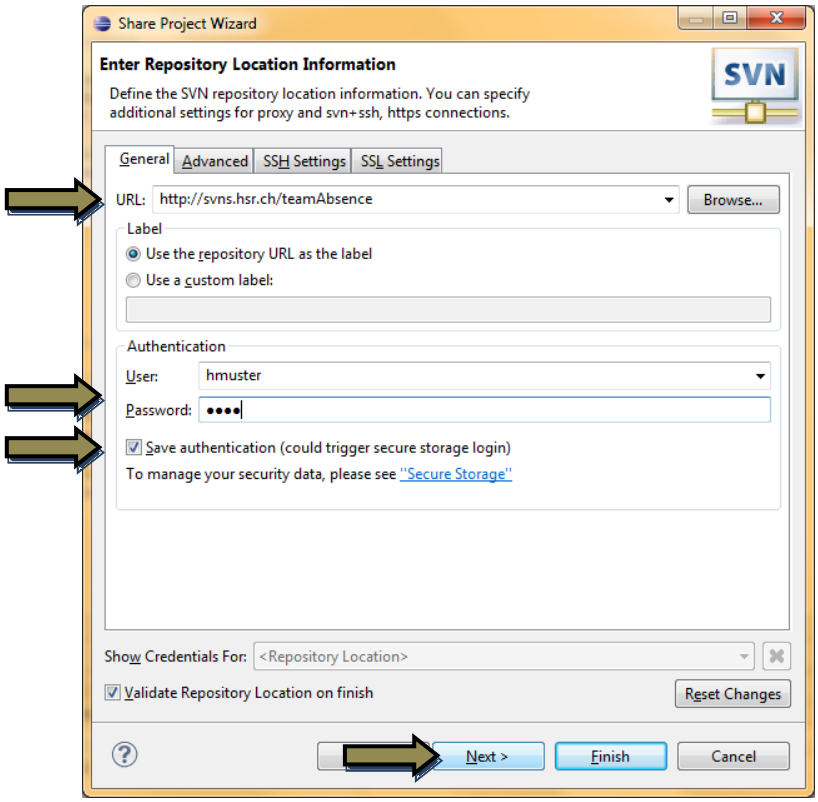


Abbildung 101: Anleitung SVN Eclipse Integration: Share Project Wizard 1

5) Anschliessend geben Sie die URL ein wo der Source Code gespeichert werden soll. Am besten ein Unterordner in Ihrem Repository, z.B. src, und klicken Sie anschliessend auf *Finish*.

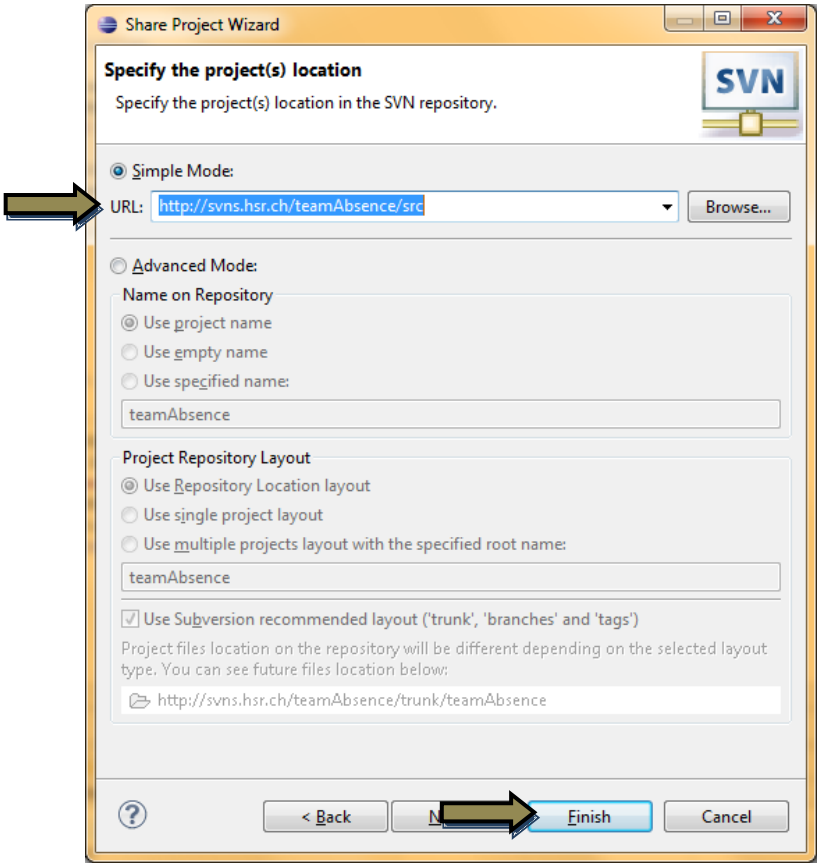


Abbildung 102: Anleitung SVN Eclipse Integration: Share Project Wizard 2

6) Führen Sie anschliessend den Commit aus.

Somit ist das Projekt nun im Repository eingecheckt und kann verwendet werden.

8.1.2.2 Projekt integrieren

Es geht hier darum ein geteiltes Projekt zu integrieren (bei anderen Teammitgliedern). Dazu muss zuerst ein Projekt geteilt werden. Die folgenden Schritte dienen dazu:

1) Wählen Sie um ein neues Projekt zu erstellen im *New Wizard* den Punkt *Other* aus und im folgenden Bildschirm *Project from SVN* und klicken Sie anschliessend auf *Next*.

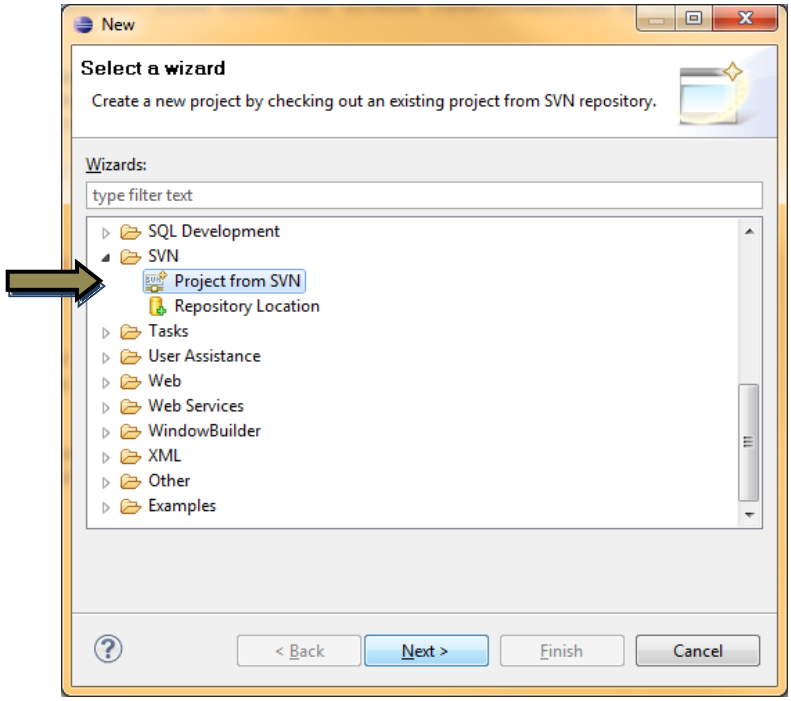


Abbildung 103: Anleitung SVN Eclipse Integration: Share Project Integration

2) Geben Sie die URL folgendermassen ein:
svn://svns.hsr.ch/<repository-name>

Geben Sie Ihren persönlichen Benutzer/Passwort ein und klicken Sie anschliessend auf *Next*.

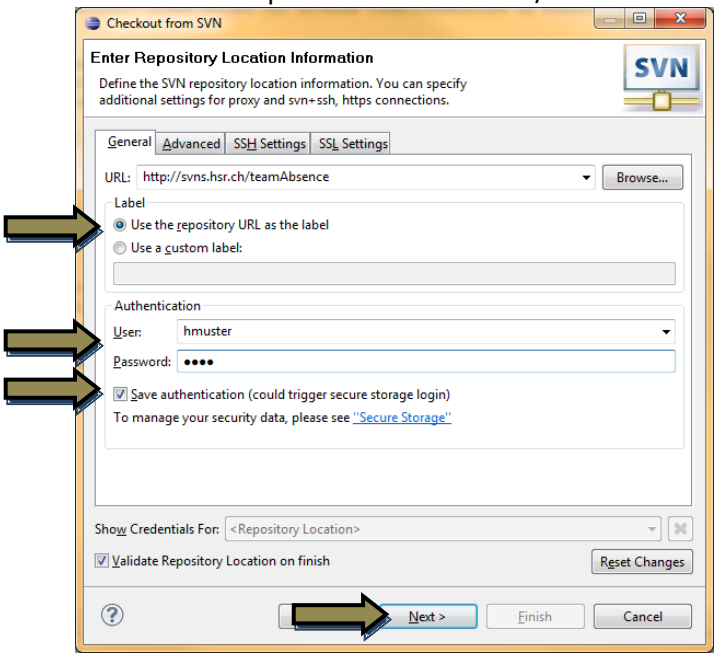


Abbildung 104: Anleitung SVN Eclipse Integration: Share Project Checkout 1

3) Geben Sie die URL zum Ordner ein, wo sich der Source Code befindet, z.B. src., und klicken Sie anschliessend auf *Finish*.

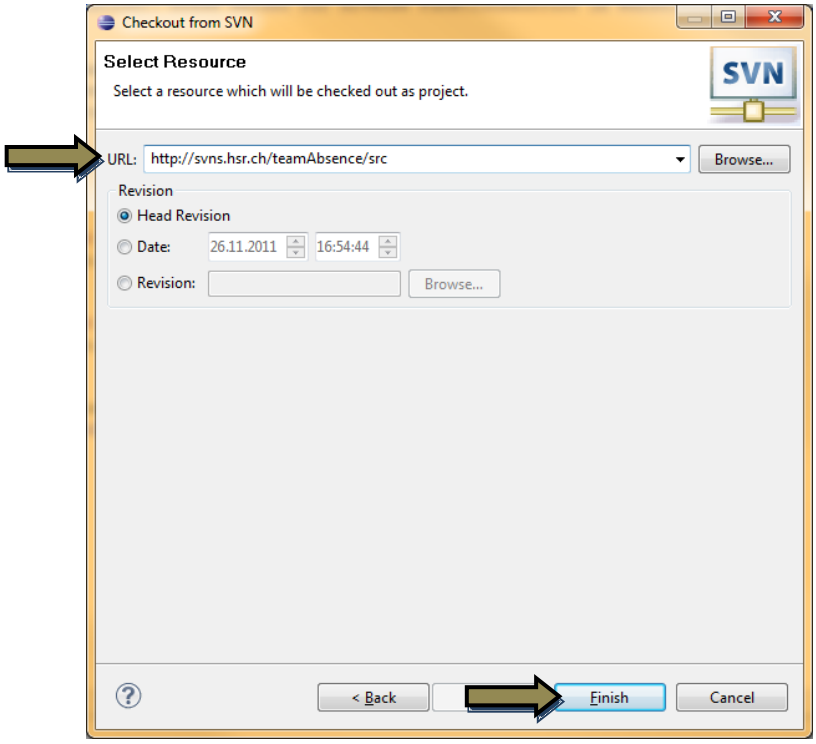


Abbildung 105: Anleitung SVN Eclipse Integration: Share Project Checkout 2

4) Checken Sie nun das Projekt so aus wie Sie es gerne in Eclipse hätten und klicken Sie auf *Finish*.

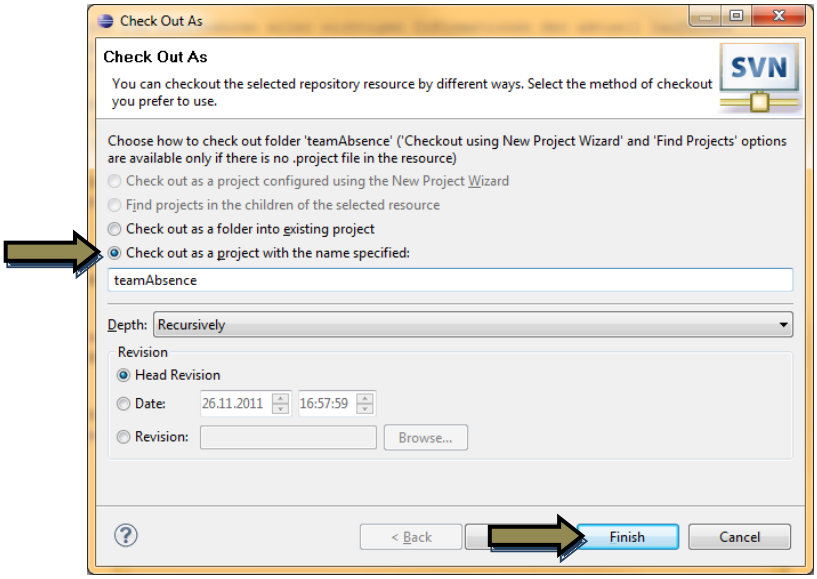


Abbildung 106: Anleitung SVN Eclipse Integration: Share Project Checkout 3

Das Projekt kann nun verwendet werden und ist mit dem Repository verbunden.

8.1.3 Anwendungstipps

Die Anwendung ist relativ selbsterklärend. Mit einem Rechtsklick auf die entsprechenden Ordner oder die entsprechende Datei, z.B. mit gemachten Änderungen, kann im Kontextmenü die gewünschte Funktion gewählt werden.

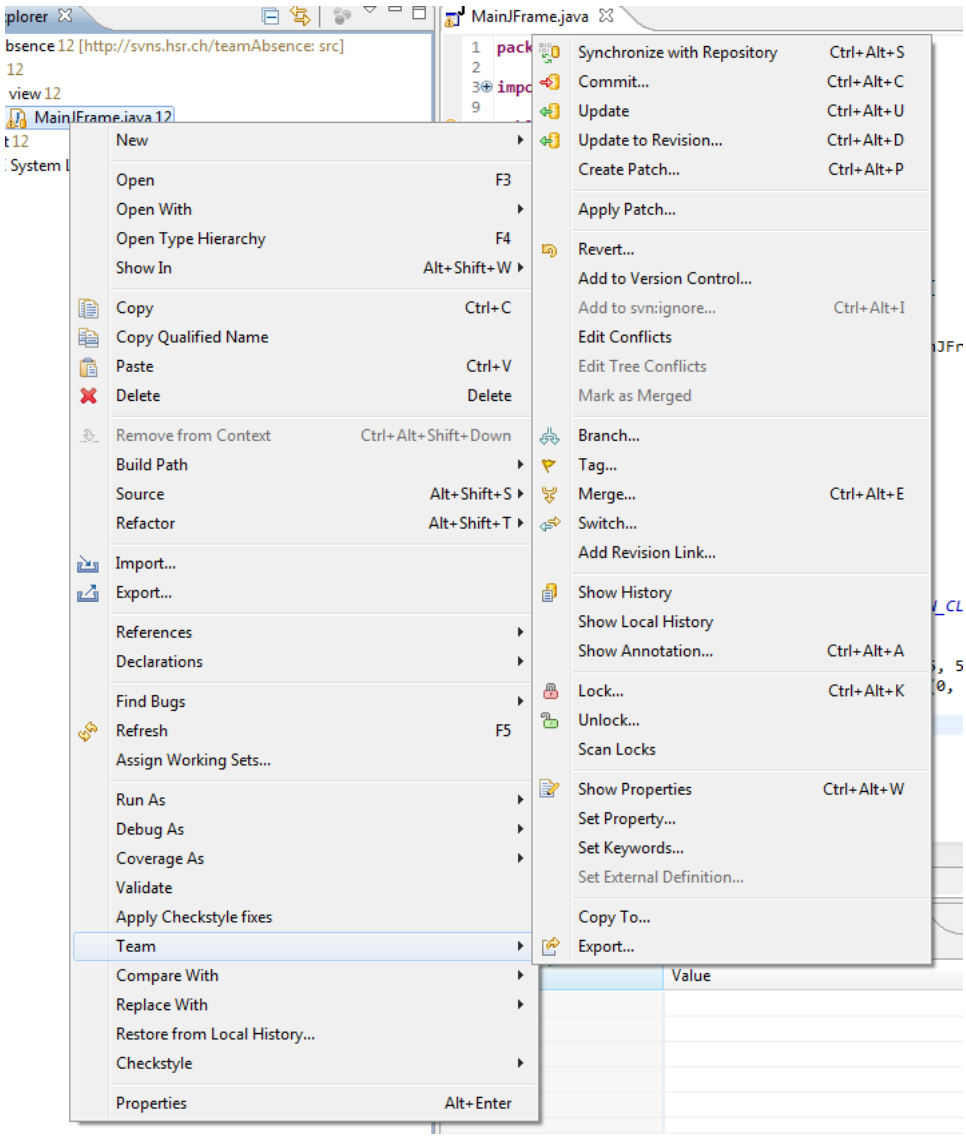


Abbildung 107: Anleitung SVN Eclipse Integration: Share Project Kontextmenü

8.2 Git (Eclipse Integration)

8.2.1 Installation

Es ist keine Installation nötig, in der neusten Version sollte Git bereits integriert sein. Falls dies nicht der Fall ist, finden Sie Informationen dazu unter http://eclipse.org/egit/.

8.2.2 Setup

8.2.2.1 Repository hinzufügen

Um das Repository von Git zu verwenden, muss es zuerst in Eclipse hinzugefügt werden. Es ist wichtig, dass Sie zuerst das Repository mit TortoiseGit oder einem ähnlichen Tool lokal geklont haben. Anschliessend führen Sie die folgenden Schritte aus:

1) Öffnen Sie die *Git Repositories* Perspektive unter *Window > Open Perspecitve > Other…*.

2) Wählen Sie *Add* aus um das voran erstellte und lokal geklonte Repository zu integrieren.

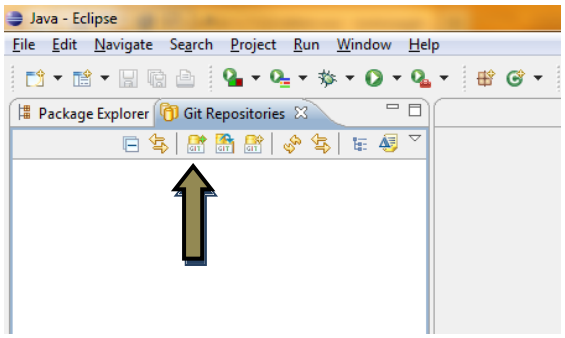


Abbildung 108: Anleitung Git Eclipse Integration: Git Repository View

3) Geben Sie das lokal geklonte Verzeichnis an und klicken Sie auf *Search*. Anschliessend wählen Sie es aus und klicken auf *OK*.

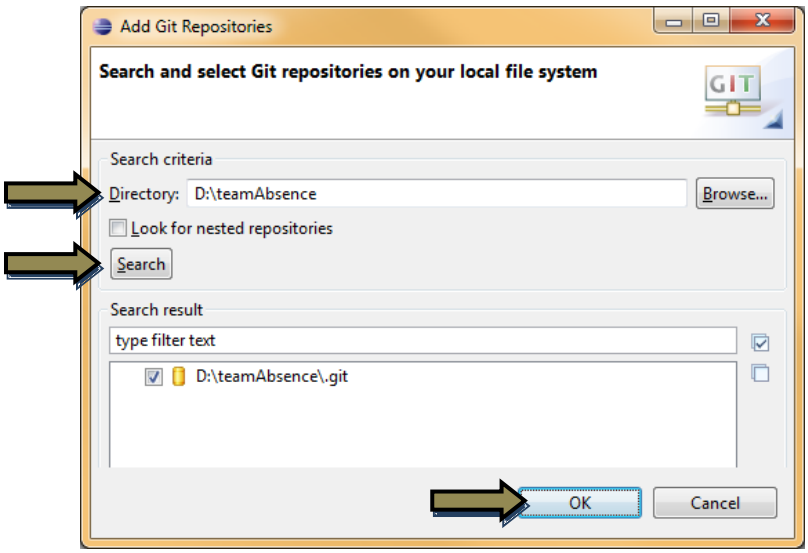


Abbildung 109: Anleitung Git Eclipse Integration: Add Git Repository

Das Repository ist nun auch in Eclipse verfügbar.

8.2.2.2 Setup Projekt / Projekt teilen

Es geht hier darum ein Projekt zu erstellen und dieses für andere Teammitglieder verfügbar zu machen. Die folgenden Schritte dienen dazu:

1) Zuerst muss Kapitel 2.1 durchgeführt werden. Danach erstellen Sie ein neues Projekt mit den entsprechenden Source Ordnern (src, test).

WICHTIG: Achten Sie darauf, dass diese Source Folder nicht leer sein dürfen! Diese führen sonst zu Kompilierfehlern bei anderen Teammitgliedern.

2) Anschliessend klicken Sie auf dem erstellten Projekt *Team > Share Project…*.

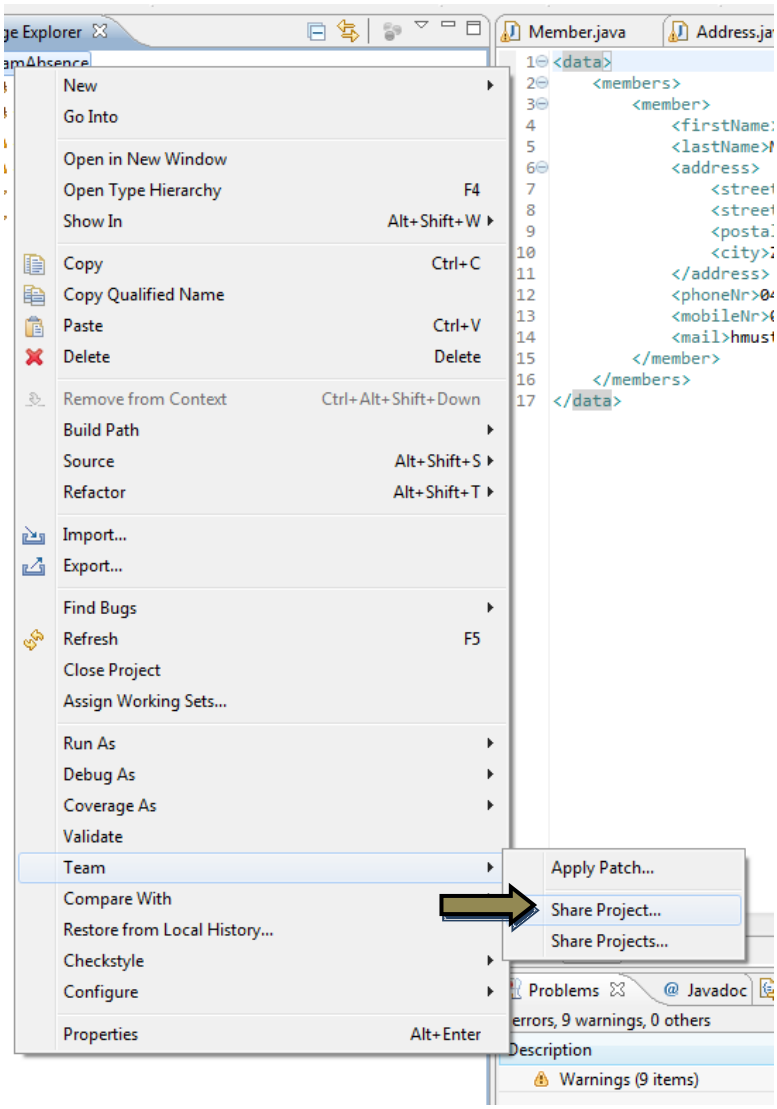


Abbildung 110: Anleitung Git Eclipse Integration: Team Kontextmenü Share

3) Wählen Sie im folgenden Fenster *Git* aus und klicken Sie auf *Next*.

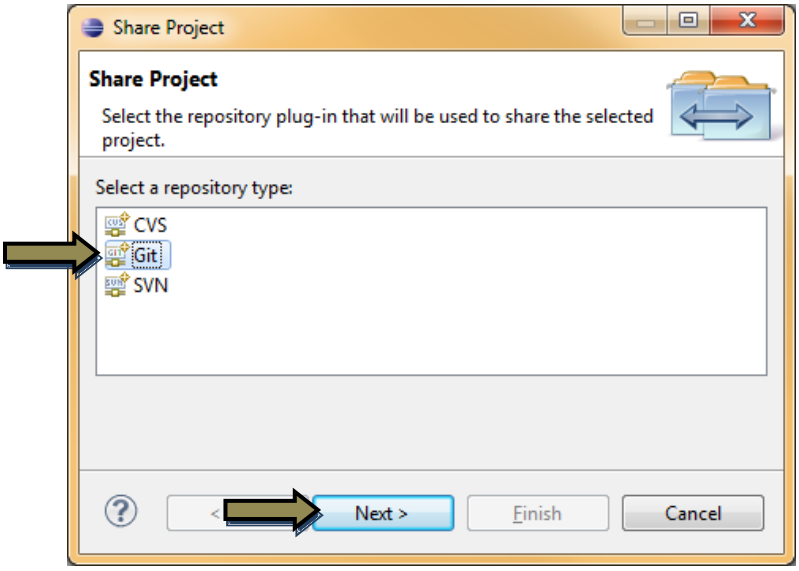


Abbildung 111: Anleitung Git Eclipse Integration: Share Git Project

4) Wählen Sie Ihr zuvor integriertes Repository aus, geben Sie einen allfälligen Source Ordner an und klicken Sie auf *Finish*.

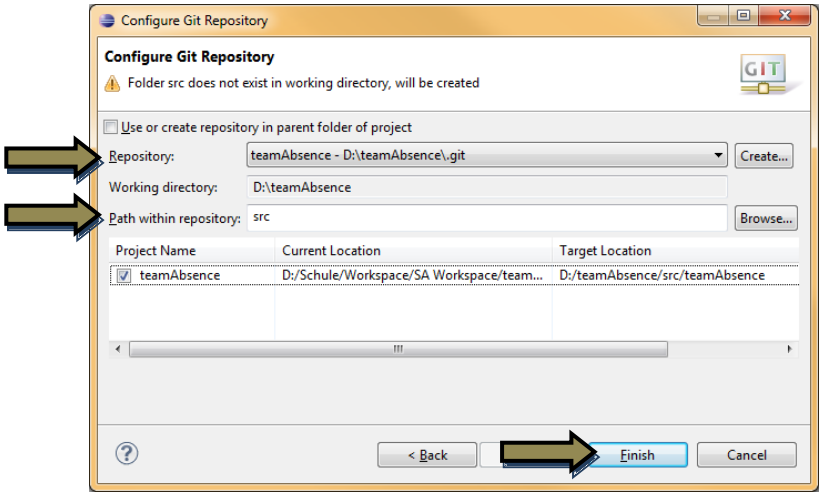


Abbildung 112: Anleitung Git Eclipse Integration: Repository Config

5) Nun folgt noch ein Commit des Projektes um die Struktur bis anhin hochzuladen, dazu werden Sie aufgefordert den Benutzername und dessen E-Mail Adresse anzugeben. Klicken Sie zum fortfahren anschliessend auf *OK*.

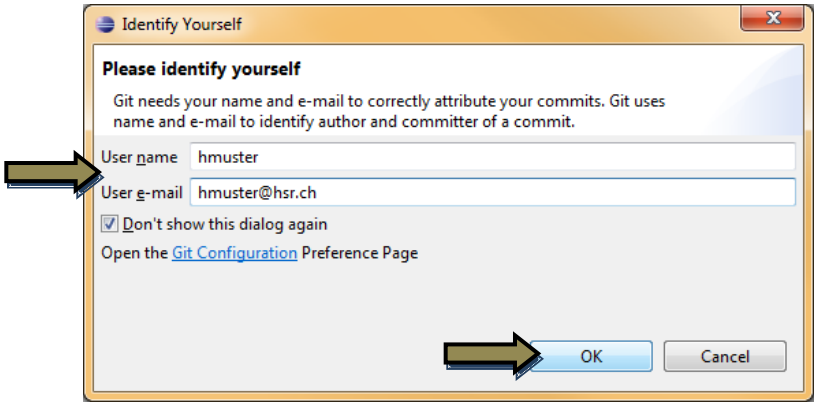


Abbildung 113: Anleitung Git Eclipse Integration: Authentifizierung

6) Wählen Sie für den Commit noch die entsprechenden Files aus und führen Sie ihn aus.

7) Da dieser Commit nur lokal ist muss noch ein Push gemacht werden.

Somit ist nun das Projekt im Repository eingecheckt und kann verwendet werden.

8.2.2.3 Projekt integrieren

WICHTIG: Es geht hier darum ein geteiltes Projekt zu integrieren (bei anderen Teammitgliedern). Dazu muss zuerst ein Projekt geteilt werden. Nachdem dieses Projekt von einem anderen Teammitglied geteilt wurde, müssen die anderen Teammitglieder sich lokal zuerst die neuste Version herunterladen (Pull). Anschliessend können die nachfolgenden Schritte durchgeführt werden:

1) Im *Import* Menü wählen Sie unter *Git* den Punkt *Projects from Git* aus und klicken auf *Next*.

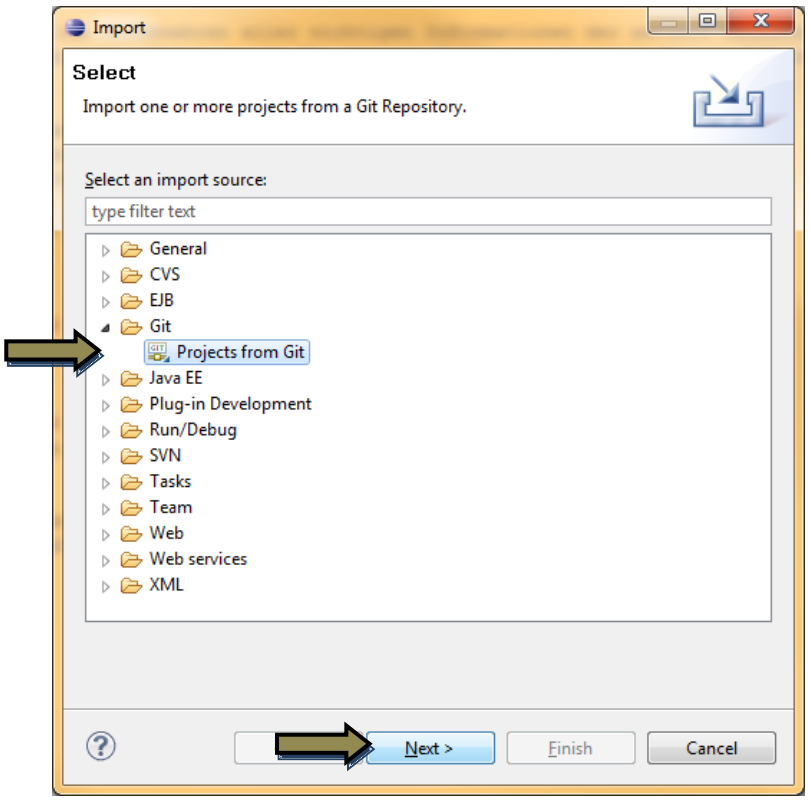


Abbildung 114: Anleitung Git Eclipse Integration: Import

2) Wählen Sie das vorhin integrierte Repository aus und klicken Sie auf *Next*.

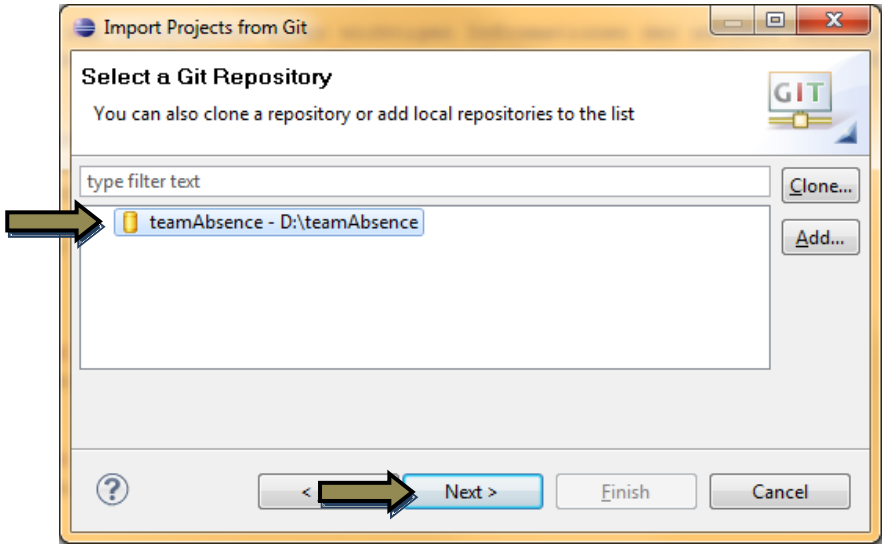


Abbildung 115: Anleitung Git Eclipse Integration: Import from Git Wizard 1

3) Wählen Sie anschliessend den Source Ordner des Projektes und das darin beinhaltende Projekt aus und klicken Sie auf *Next*.

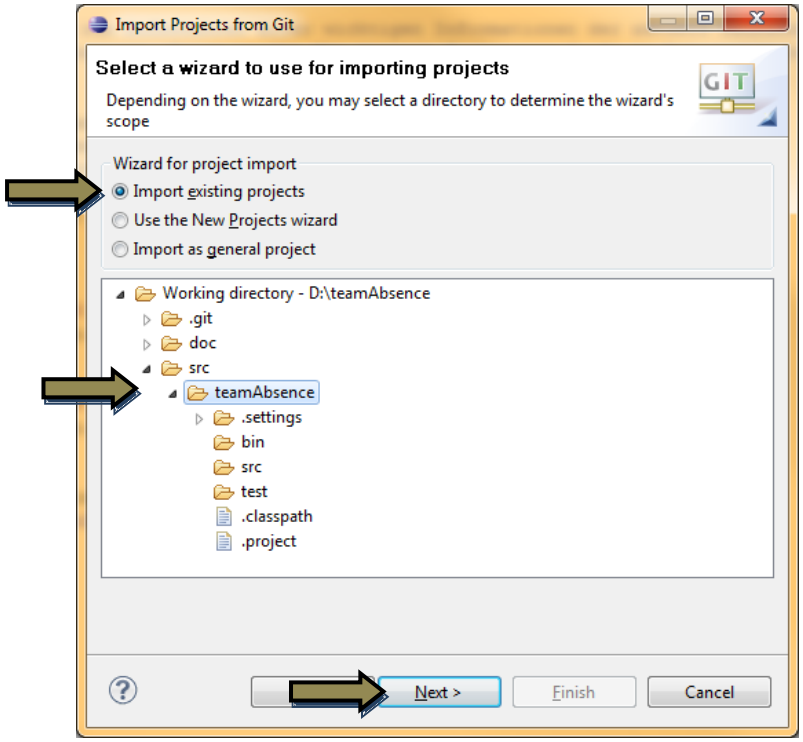


Abbildung 116: Anleitung Git Eclipse Integration: Import from Git Wizard 2

4) Klicken Sie abschliessend auf *Finish*.

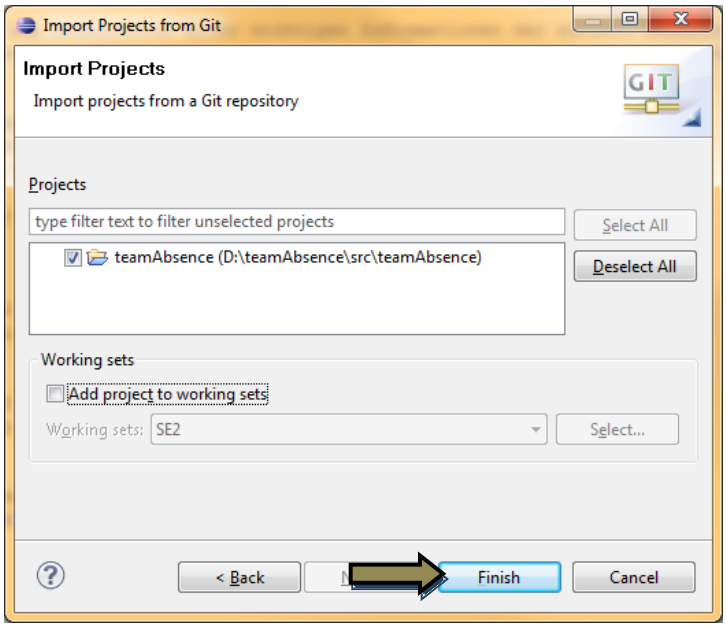


Abbildung 117: Anleitung Git Eclipse Integration: Import from Git Wizard 3

Das Projekt ist nun in Eclipse integriert und kann verwendet werden.

8.2.3 Anwendungstipps

Die Anwendung ist relativ selbsterklärend. Mit einem Rechtsklick auf die entsprechenden Ordner oder die entsprechende Datei, z.B. mit gemachten Änderungen, kann im Kontextmenü die gewünschte Funktion gewählt werden.

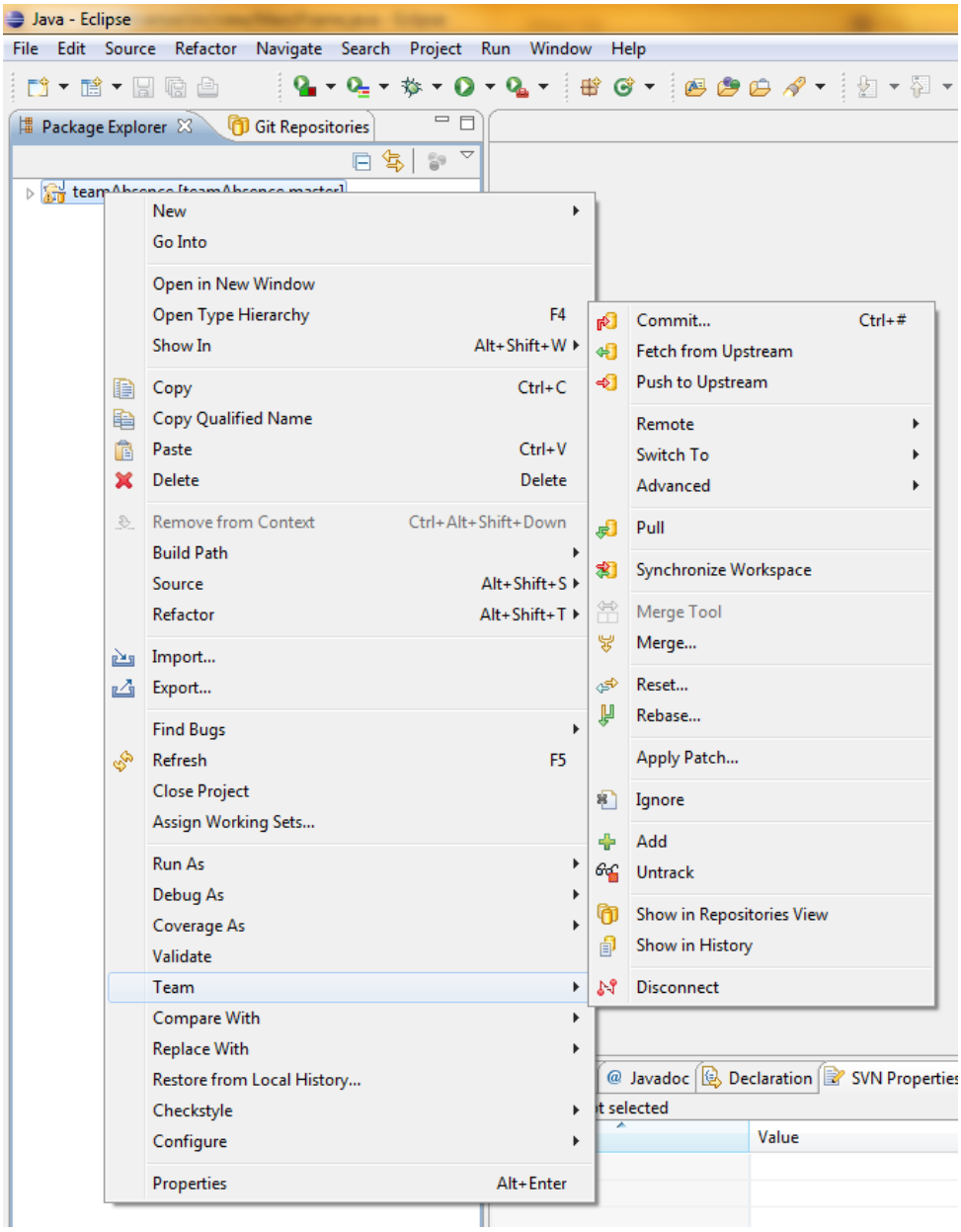


Abbildung 118: Anleitung Git Eclipse Integration: Team Kontextmenü

8.3 SVN Repository

8.3.1 Installation

Eine Installation ist nicht notwendig, da ein SVN Server von der Schule bereits angeboten wird.

8.3.2 Setup

1) Öffnen Sie folgende URL: https://svns.hsr.ch
Benutzername und Passwort entsprechen dem HSR Account.

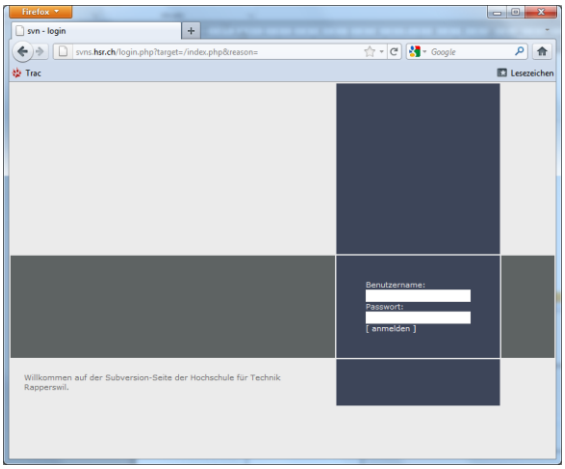


Abbildung 119: Anleitung SVN Repository: Login

2) Um ein neues Projekt zu erstellen klicken Sie auf *Projekt erstellen*.

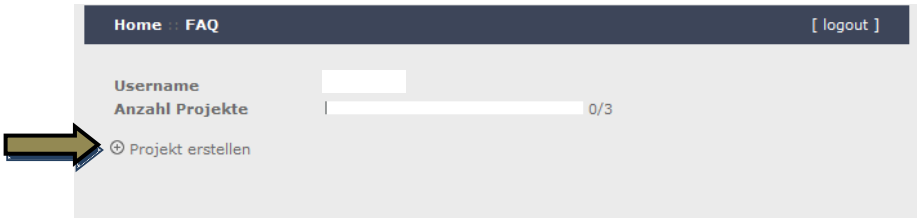


Abbildung 120: Anleitung SVN Repository: Projekt erstellen 1

3) Geben Sie die entsprechenden Informationen zu Ihrem Projekt an und klicken Sie auf *hinzufügen*.

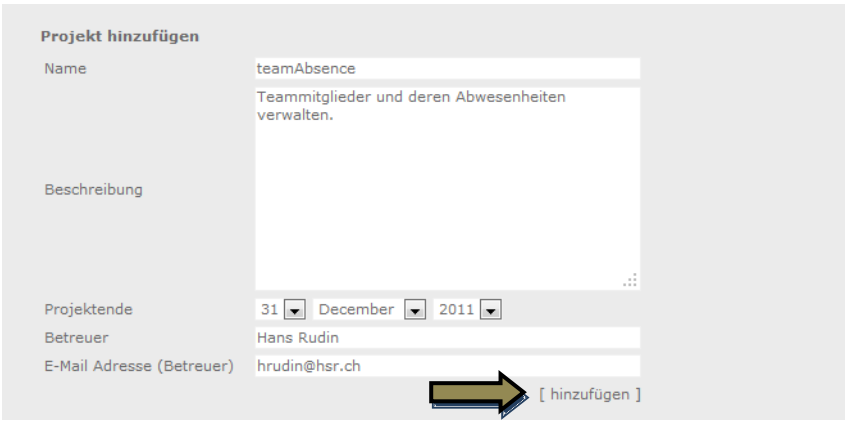


Abbildung 121: Anleitung SVN Repository: Projekt erstellen 2

4) Das Projekt ist nun erstellt. Alle Benutzer (Teammitglieder) können/müssen über die Schaltfläche *neuen Benutzer hinzufügen* dem Projekt hinzugefügt werden.

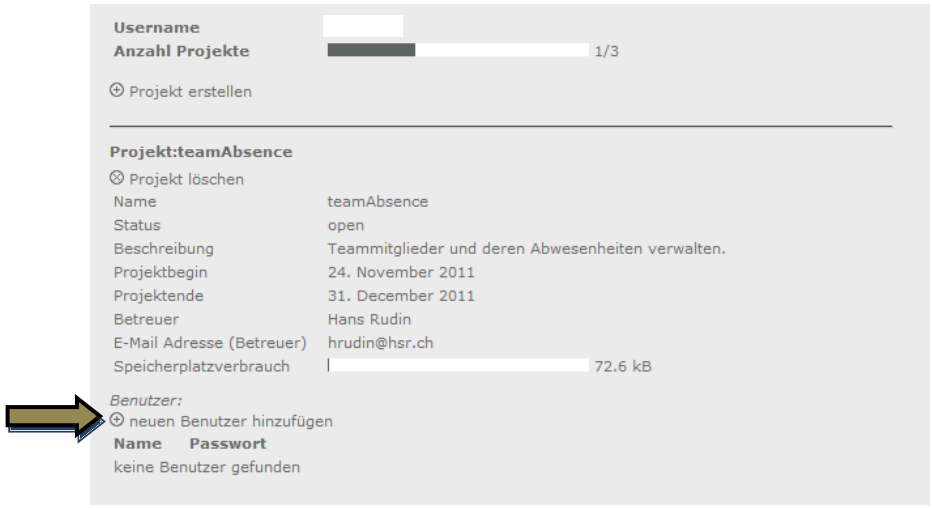


Abbildung 122: Anleitung SVN Repository: Projektübersicht

5) Durch Eingabe eines Kürzels des Benutzers und eines Passworts (werden später für die Verwendung benötigt, Kürzel sollte kein Trennzeichen enthalten) werden einzelne Benutzer erstellt.

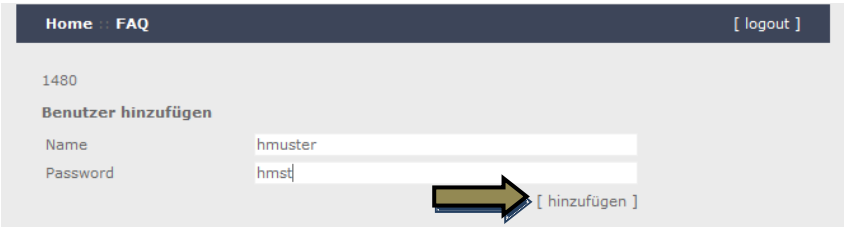


Abbildung 123: Anleitung SVN Repository: Benutzer erstellen

6) Erstellen Sie so für jedes Teammitglied einen eigenen Benutzer.

7) Das Projekt ist nun erstellt und könnte unter folgender URL verwendet werden: svn://svns.hsr.ch/teamAbsence (Gross-/Kleinschreibung beachten!)

8.3.3 Anwendungstipps

Informationen finden Sie unter https://svns.hsr.ch/faq.php oder über die HSR Webseite.

Allgemeine Anwendungstipps finden Sie in den anderen Anleitungen zu SVN (siehe AnleitungTortoiseSVN.pdf oder AnleitungSVNEclipse.pdf).

8.4 GIT Repository

8.4.1 Installation

Ist nicht notwendig, da Git bereits auf dem virtuellen Server eingerichtet wurde.

8.4.2 Setup

In diesem Abschnitt wird gezeigt, wie es Ihnen möglich ist auf dem Server ein Git Repository einzurichten.

1) Verbinden sie sich mit dem Konsoleninterface des virtuellen Servers und loggen sich dort als root ein. Die Anleitung hierzu, sollten Sie per-Mail von den Informatikdiensten der HSR erhalten haben. Falls nicht finden Sie die Anleitung auf dem Skripteserver unter: http://skripte.hsr.ch/Informatik/Fachbereich/SA-DA-BA/VirtualInfrastructureGuide.pdf

2) Führen Sie ein Update aller installierten Pakete durch.
   a. `apt-get upgrade`

3) Für jedes Projektmitglied einen User mit einem Passwort erstellen mit.
   a. `adduser <username>`
   b. `passwd <username> <password>`

4) Die erstellten User der Gruppe git-users hinzufügen.
   a. `usermod -G git-users <username>`

5) Ins Verzeichnis /var/gitrepo wechseln.
   a. `cd /var/gitrepo`

6) Ein neues Repository erstellen.
   a. `mkdir <repositoryname>.git`
   b. `cd <repositoryname>.git`

7) Nun wird das GIT-Repository initialisiert (- -share damit alle Zugriff erhalten, - -bare bedeutet kein working directory sondern Repository).
   a. `git init --bare --share=group`
   b. Dieses ist nun erreichbar unter:
   `ssh://<user>@<server-name>/var/gitrepo/<repository-name>.git`

8) Die Clients sollten nun in der Lage sein, das Git Repository zu klonen um es lokal verwenden zu können.

8.4.3 Anwendungstipps

Informationen finden Sie unter http://git-scm.com/.

Allgemeine Anwendungstipps finden Sie in den anderen Anleitungen zu Git (siehe AnleitungTortoiseGit.pdf oder AnleitungGitEclipse.pdf).

8.5 Inbetriebnahme virtueller Server

8.5.1 Verbindungsaufbau

Verbinden sie sich mit dem Konsoleninterface des virtuellen Servers und loggen sich dort als root ein. Die Anleitung hierzu, sollten Sie per-Mail von den Informatikdiensten der HSR erhalten haben. Falls nicht finden Sie die Anleitung auf dem Skripteserver unter:

http://skripte.hsr.ch/Informatik/Fachbereich/SA-DA-BA/VirtualInfrastructureGuide.pdf

Zum Aufbau der Verbindung empfehlen wir (für Windows) das Terminal-Programm PuTTY.

8.5.2 RSA Schlüssel Änderung

Die RSA-Keys (private und public) sind aufgrund des Klon-Prozesses nicht mehr einzigartig und müssen somit erneuert werden.

In diesem Abschnitt wird gezeigt, wie es Ihnen möglich ist auf dem Server die Host Keys zu erneuern.

1) Nach dem erfolgreichen Verbindungsaufbau wechseln Sie ins Verzeichnis /usr/local/scripts/changekey
   a. `cd /usr/local/scripts/changekey`

2) Führen Sie nun das Script change_rsa.sh aus.
   a. `./change_rsa.sh`

3) Starten Sie, nach Beendigung den virtuellen Server neu.
   a. `reboot`

Hinweis: Beim nächsten Verbindungsaufbau wir PuTTY ihnen eine Sicherheitswarnung anzeigen, welche Sie bestätigen können.

8.5.3 Aktualisierung der Programme

Führen Sie ein Update aus, um die installierten Programme auf den neusten Stand zu bringen:

1) Hierfür führen Sie folgende befehle aus:

- `apt-get update`
- `apt-get upgrade`
- `aptitude update`
- `aptitude safe-upgrade`

Vorsicht: Bestätigungsaufforderungen genau lesen, ansonsten können gewisse Programmen nicht mehr wie gewünscht funktionieren.

Beispielsweise Jenkins wäre nach einer Bestätigung (Y) dieser Meldung nicht mehr über das Webinterface erreichbar.

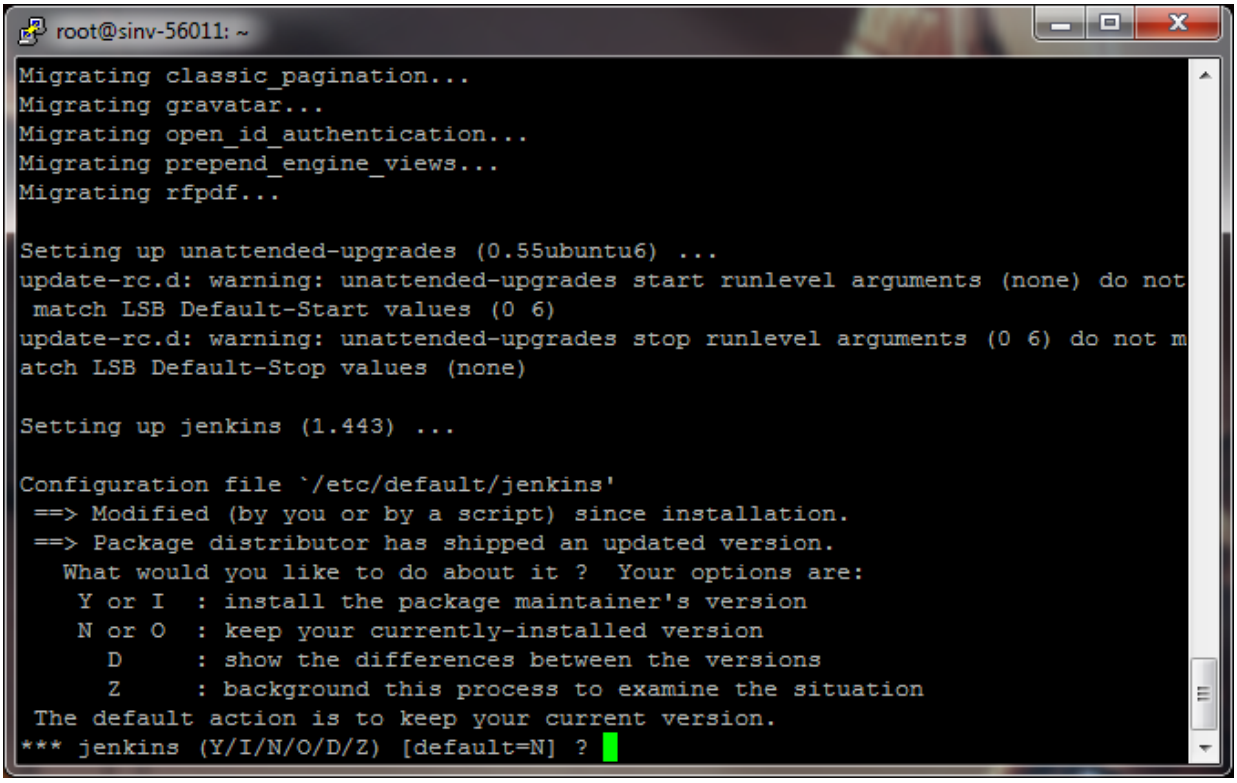


Abbildung 124: Anleitung Inbetriebnahme Server: Update Warnung

8.6 Redmine

8.6.1 Installation

Ist nicht notwendig, da Redmine auf dem virtuellen Server bereits eingerichtet ist.

8.6.2 Setup

8.6.2.1 Login

1) Gehen Sie auf die Startseite Ihres virtuellen Servers (z.B. sinv-00100.edu.hsr.ch) und wählen Sie dort *Redmine* aus.

2) Mit dem Mitgliedsnamen admin und dem Passwort admin können Sie sich nun einloggen.

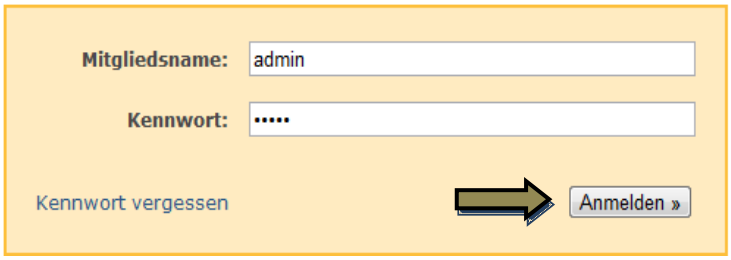


Abbildung 125: Anleitung Redmine: Login

8.6.2.2 Benutzer erstellen

1) Nach der erfolgreichen Anmeldung sehen Sie eine leere Hauptseite, wählen oben *Administration* aus und klicken Sie anschliessend auf den Punkt *Benutzer*.

Abbildung 126: Anleitung Redmine: Administrationsmenü 1

2) Im folgenden Fenster können Benutzer erstellt und verwaltet werden, um einen neuen Benutzer zu erstellen klicken Sie auf *Neuer Benutzer*.

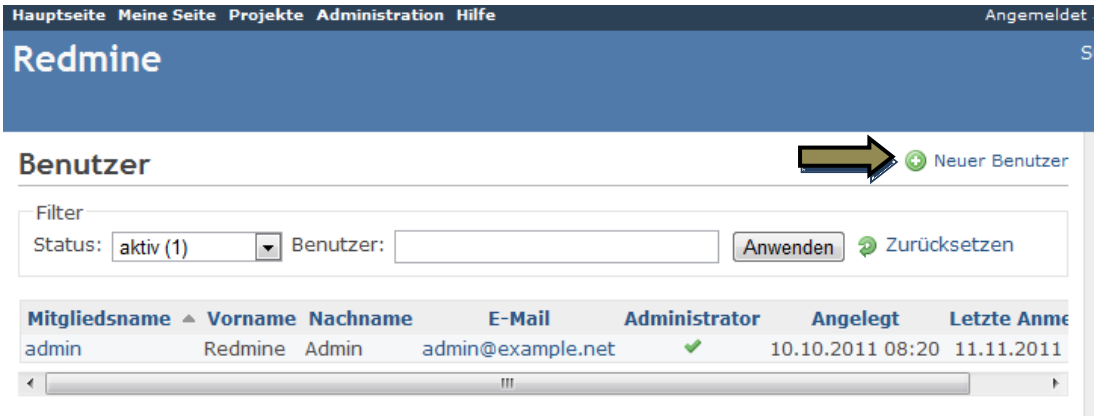


Abbildung 127: Anleitung Redmine: Benutzer Hinzufügen

3) Erfassen Sie nun die gewünschten Benutzer. Es ist nötig für jedes Projektmitglied einen eigenen Benutzer zu erfassen.

Hinweis: Es ist **kein** SMTP-Server eingerichtet, somit werden keine Benachrichtigungen per E-Mail versendet.

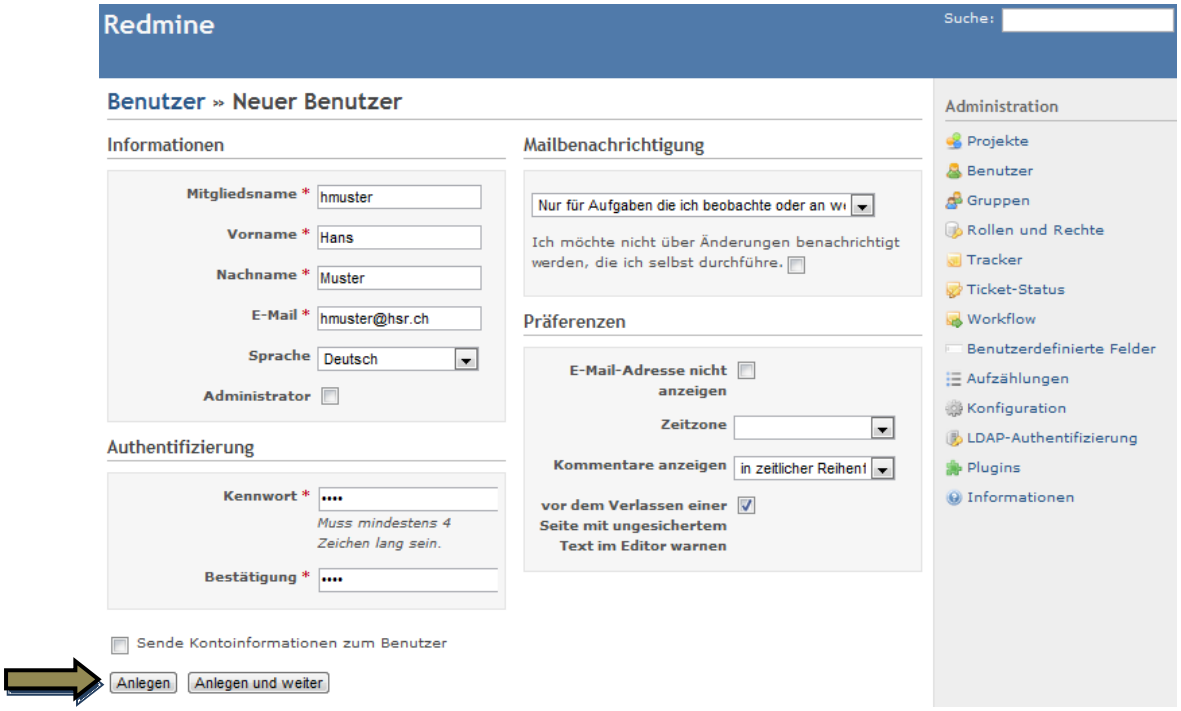


Abbildung 128: Anleitung Redmine: Neuer Benutzer

Mit *Anlegen* wird der Benutzer angelegt und Sie gelangen zur Übersicht, mit *Anlegen und weiter* können Sie gleich mehrere Benutzer anlegen.

8.6.2.3 Projekt erstellen

1) Wählen Sie nun unter *Administration* den Punkt *Projekte* aus.

Abbildung 129: Anleitung Redmine: Administrationsmenü 2

2) Um ein neues Projekt zu erstellen klicken Sie auf *Neues Projekt*.

Abbildung 130: Anleitung Redmine: Projekt hinzufügen

3) Erstellen Sie nun Ihr Projekt.

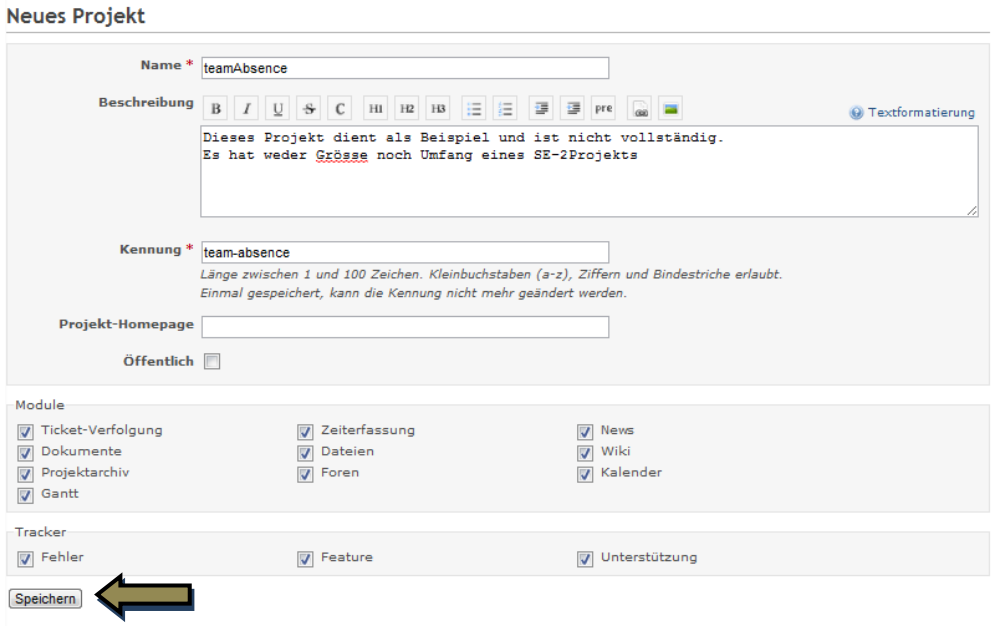


Abbildung 131: Anleitung Redmine: Neues Projekt

4) Nach dem Speichern gelangen Sie zur folgenden Ansicht. Um Mitglieder des Projektes auszuwählen klicken Sie auf den Tab *Mitglieder*.

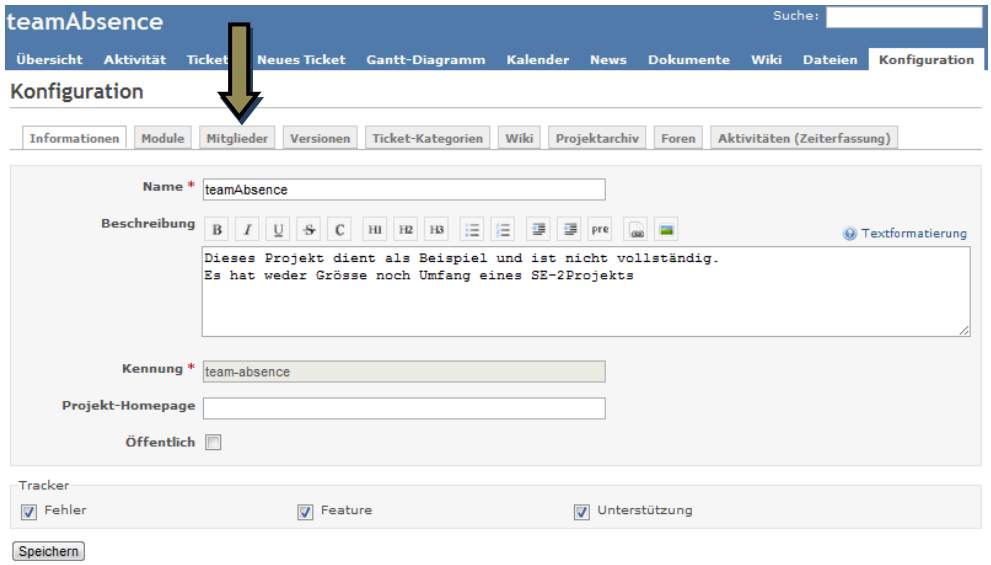


Abbildung 132: Anleitung Redmine: Projekt Informationen

5) Wählen Sie die gewünschten Mitglieder aus und ordnen Sie diese dem Projekt zu. Den Mitgliedern kann noch eine Funktion zugeteilt werden.

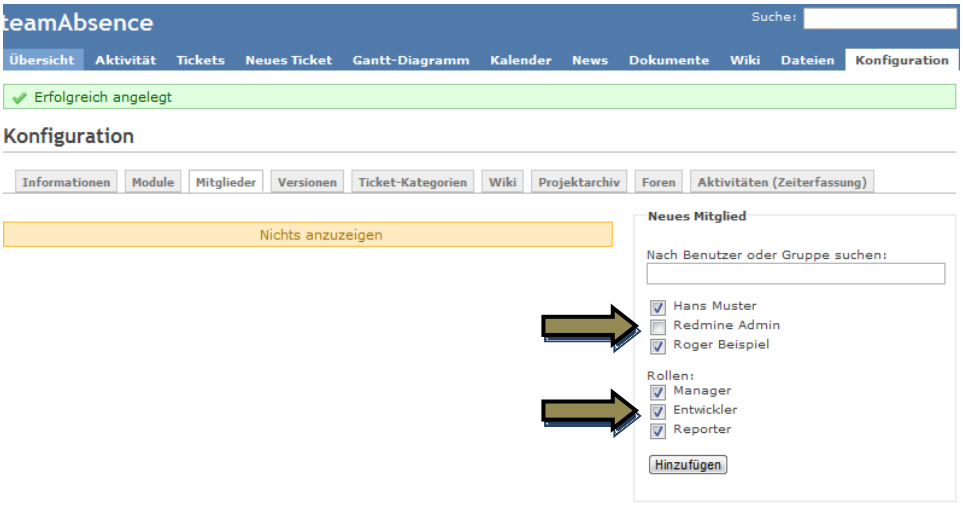


Abbildung 133: Anleitung Redmine: Projektmitglieder hinzufügen

8.6.2.4 SVN/Git Integration

Als letzten Schritt folgt die Integration von SVN oder Git. Hierzu wählen Sie den Tab *Projektarchiv*. Dort können Sie zwischen *SVN* und *Git* wählen.

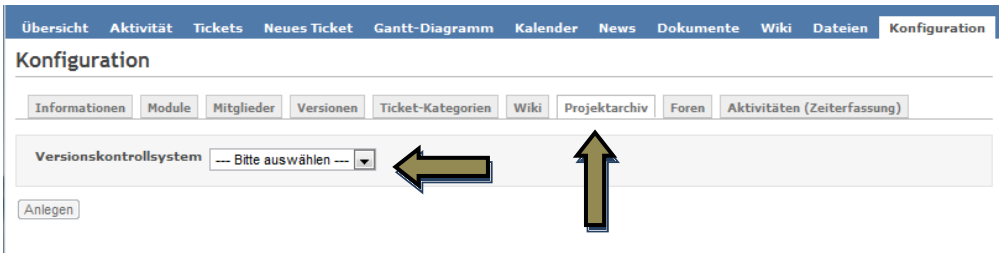


Abbildung 134: Anleitung Redmine: Projektarchiv Integration

8.6.2.4.1 Konfiguration für SVN

Geben Sie die URL des SVN Servers und die Login Daten an.
Tipp: Auf dem SVN Server ein User redmine erstellen.

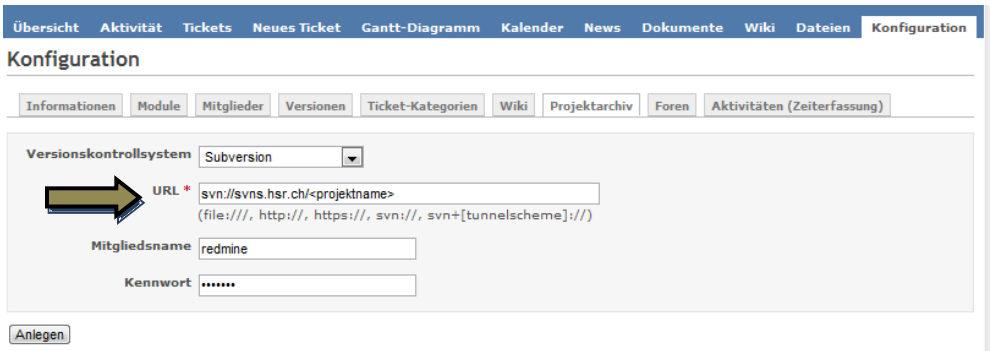


Abbildung 135: Anleitung Redmine: Projektarchiv Integration SVN

8.6.2.4.2 Konfiguration für Git

Das lokale Verzeichnis des Git-Repository auf Ihrem virtuellen Server lautet:
/var/gitrepo/<repositoryname>

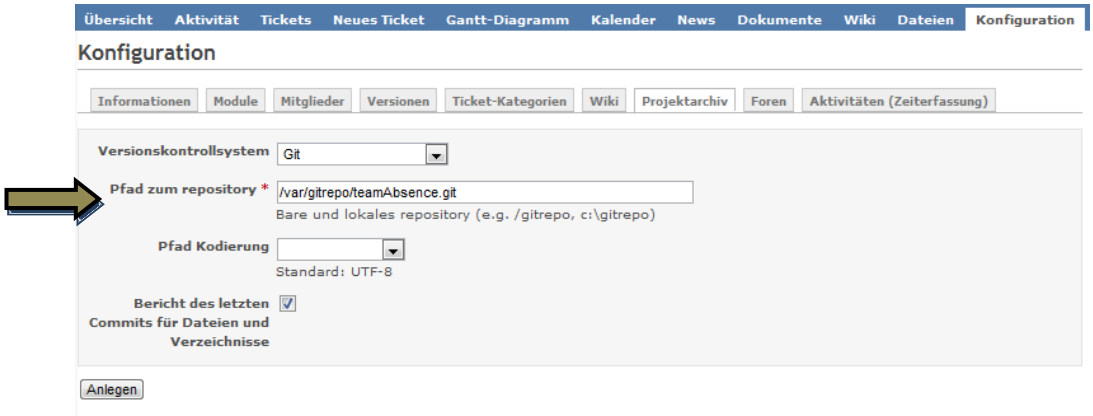


Abbildung 136: Anleitung Redmine: Projektarchiv Integration Git

8.6.3 Anwendungstipps

In diesem Kapitel werden einige Grundlegende Funktionen genauer erläutert.

Sollte Sie Probleme bei der Bedienung haben, nutzen Sie die offiziellen Hilfeseiten von Redmine: http://www.redmine.org/guide.

8.6.3.1 Iterationen erfassen

1) Um die Iterationen zu erfassen, muss in Redmine eine Version erstellt werden. Hierzu muss man als ein Administrator angemeldet sein.

2) Als nächstes muss das Projekt ausgewählt werden. Im Menü *Konfiguration* findet sich ein Tab *Versionen*. Klicken Sie dort auf *Neue Version*.

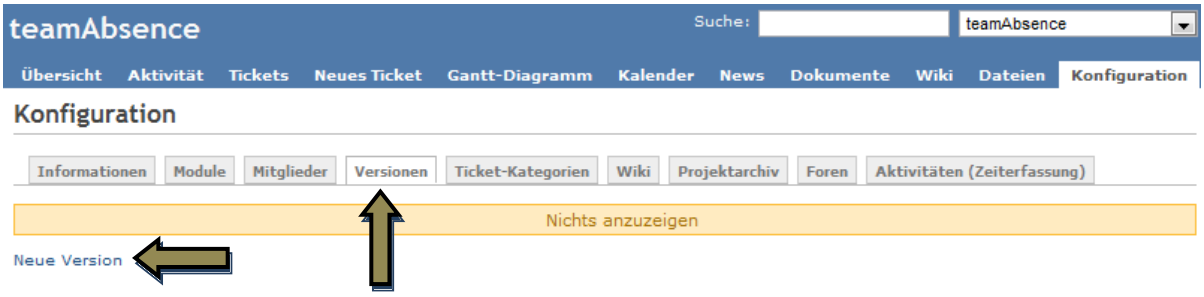


Abbildung 137: Anleitung Redmine: Version erstellen

3) Nun können Sie die Iteration (die Version) erfassen.

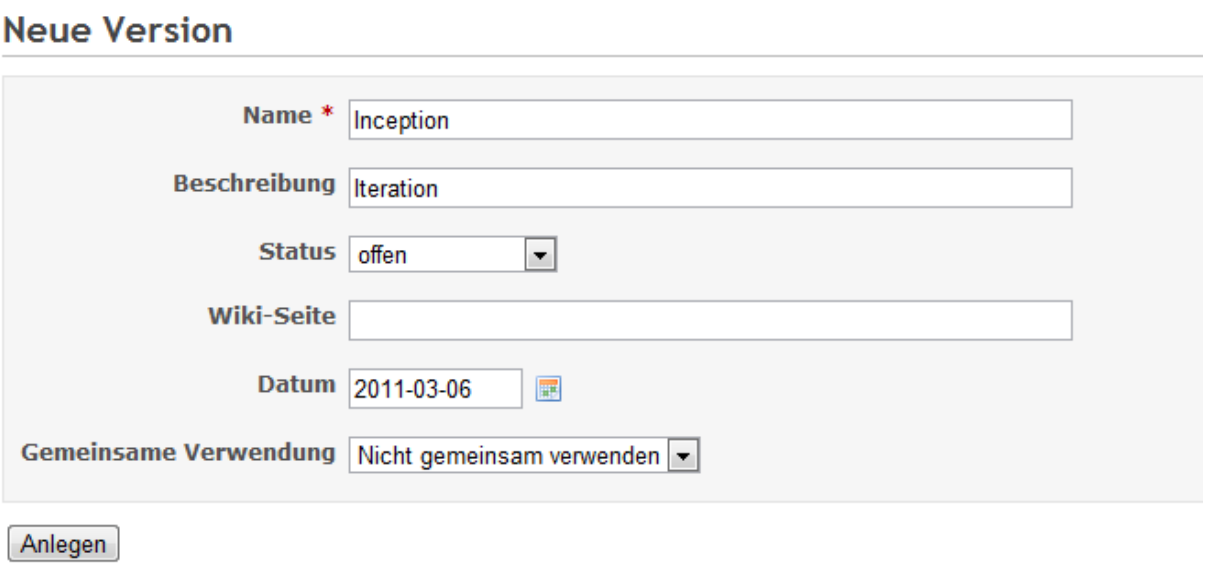


Abbildung 138: Anleitung Redmine: Neue Version

4) Dieser Iteration kann nun bei der Erfassung eines Tickets angegeben werden.
Hinweis: Versionen sind im Gant-Chart nur sichtbar, wenn sie ein Ticket enthalten.

8.6.3.2 Tickets erfassen

Als Projektmitglied angemeldet, ist es Ihnen möglich ein Ticket zu erstellen. Tickets werden dazu verwendet, die anfallenden Arbeiten zu erfassen.
Um ein Ticket zu erstellen wählen Sie den Punkt *Neues Ticket* aus.

Erfassen die nötigen Daten und klicken Sie auf *Anlegen* falls Sie noch weitere Tickets zu erstellen haben klicken Sie auf *Anlegen und weiter*.

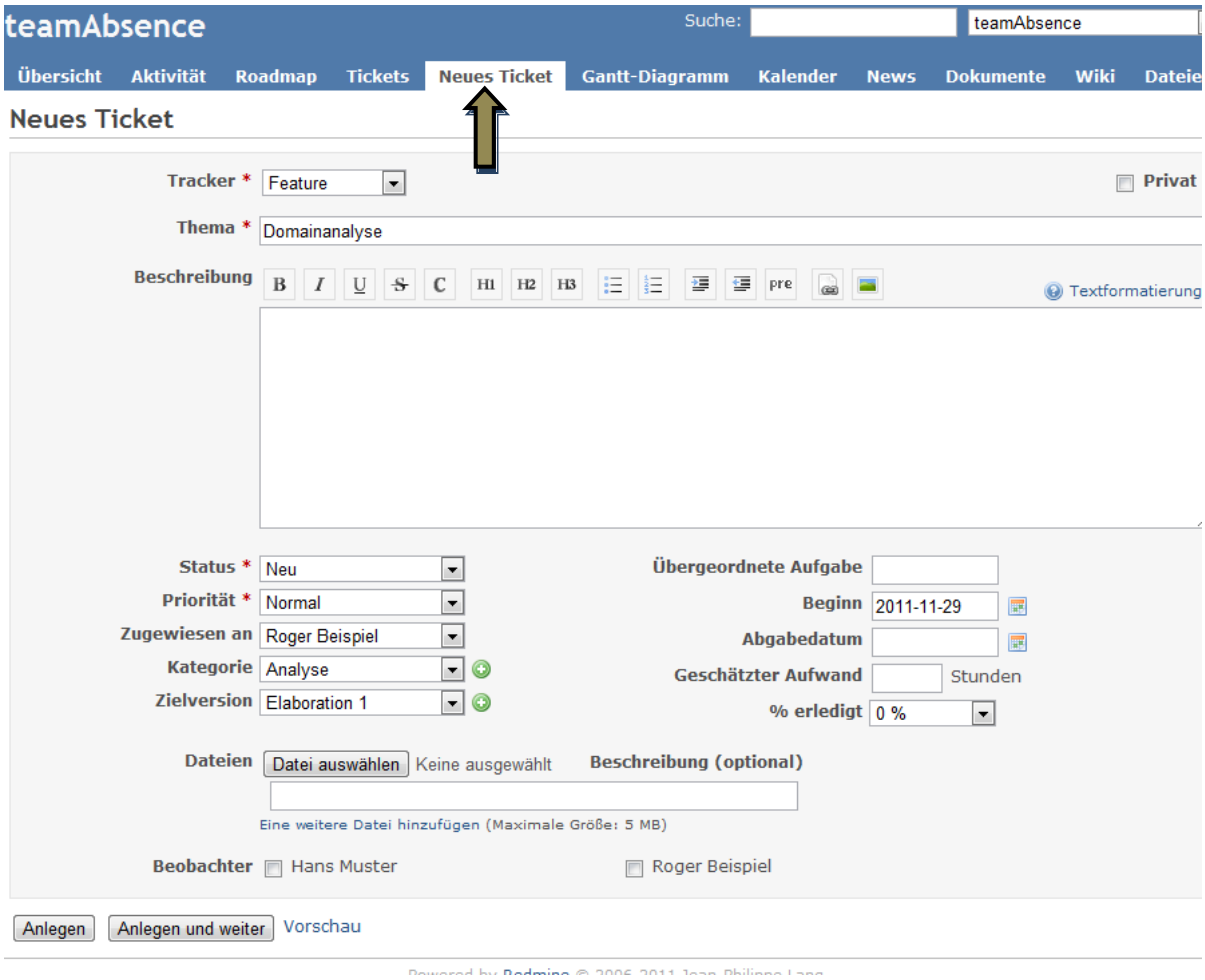


Abbildung 139: Anleitung Redmine: Neues Ticket

8.6.3.3 Aktivitäten erfassen

Mit Hilfe dieser Aktivitäten, ist es möglich die Arbeit an einem Tickets den RUP-Disziplinen zuzuordnen. Dies ermöglicht eine bessere Auswertung.

Um eine Aktivität zu erfassen, müssen Sie als Administrator angemeldet sein. Im *Administration* Menü unter dem Punkt *Aufzählung* können Aktivitäten verwaltet werden.

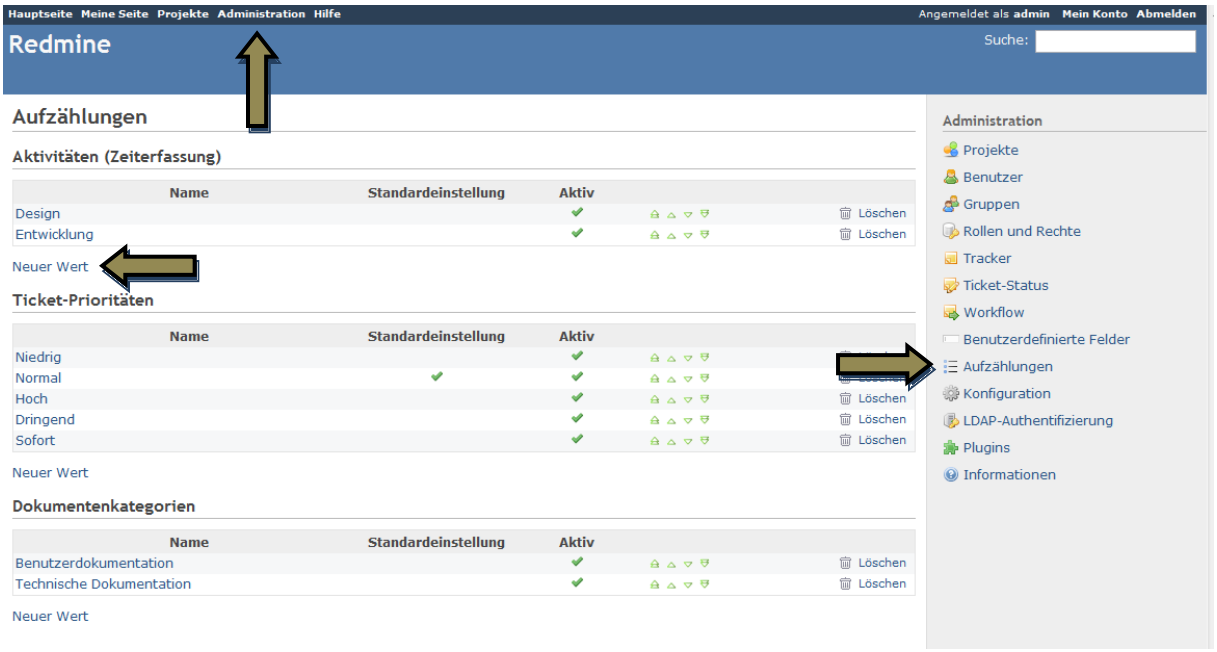


Abbildung 140: Anleitung Redmine: Aktivität hinzufügen

8.6.3.4 Zeiterfassung

1) Um die Arbeitszeiten zu erfassen, muss für die Arbeit ein Ticket erstellt sein.

Um die Zeit zu erfassen, müssen Sie das entsprechende Ticket auswählen.

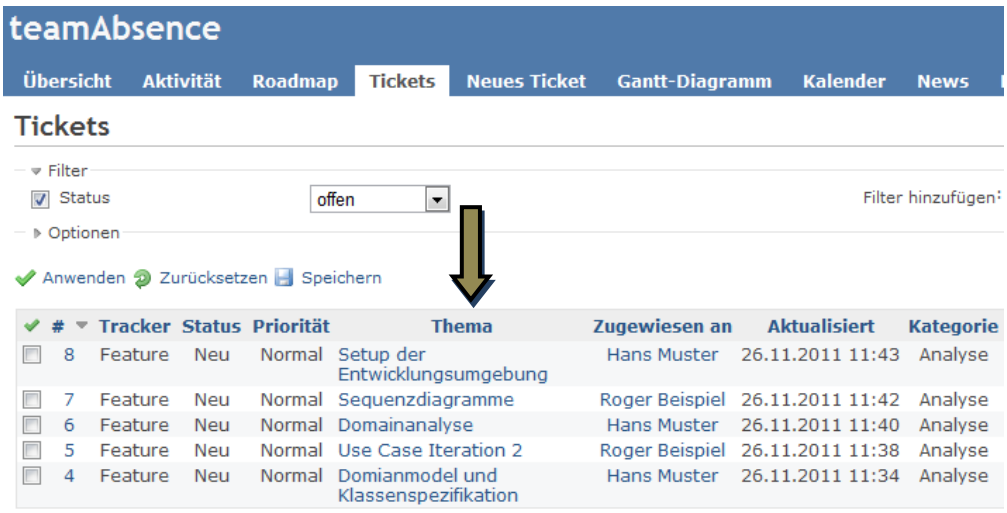


Abbildung 141: Anleitung Redmine: Ticketübersicht

2) Klicken Sie auf *Aufwand buchen*.

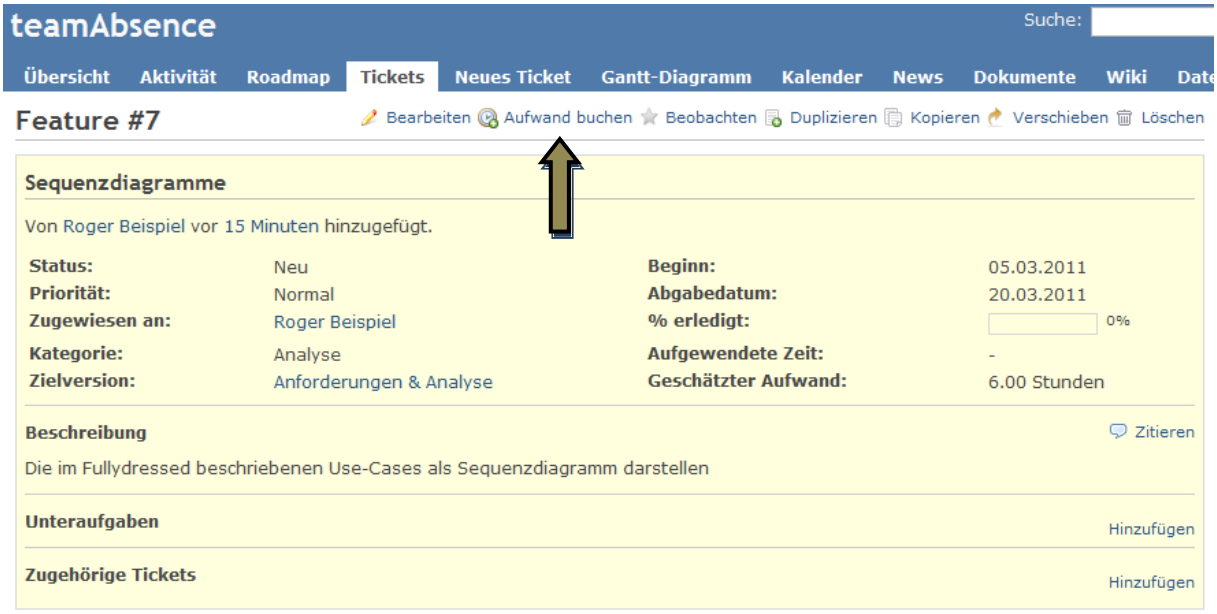


Abbildung 142: Anleitung Redmine: Ticket Detailansicht

3) Wählen Sie Datum und Aktivität, tragen Sie die aufgewendeten Stunden ein und klicken Sie *Speichern*.

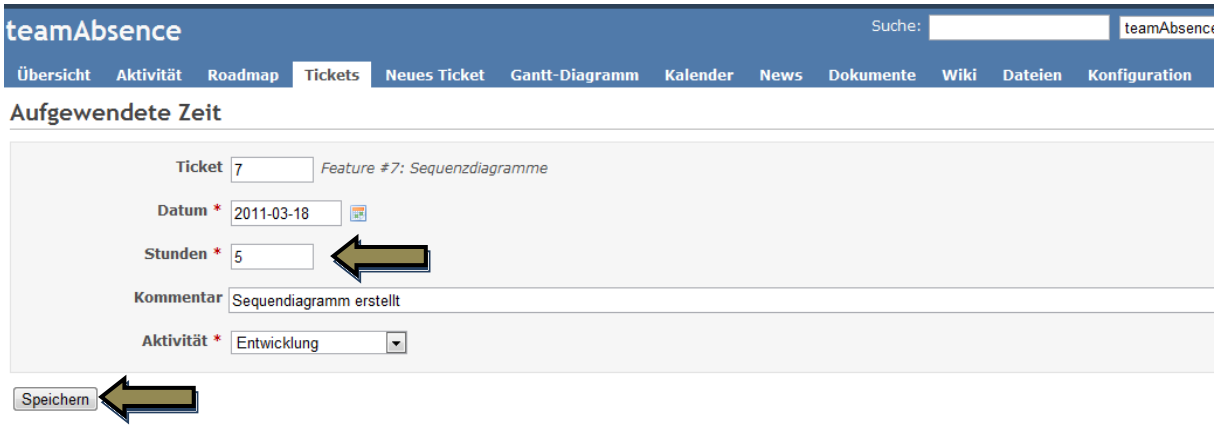


Abbildung 143: Anleitung Redmine: Aufwand buchen

8.6.3.5 Inhalte exportieren

In Redmine können Sie beinahe alles Exportieren. Achten Sie in der jeweiligen Ansicht auf folgendes:

Abbildung 144: Anleitung Redmine: Exportfunktionen

Dies befindet sich meist am linken unteren Rand.

8.7 Jenkins

8.7.1 Installation

Eine Installation ist nicht notwendig, da Jenkins auf dem virtuellen Server bereits eingerichtet ist.

8.7.2 Setup

1) Folgenden Schritte müssen bereits erledigt sein:
   a. SVN oder Git eingerichtet.

2) Gehen Sie auf die Startseite Ihres virtuellen Servers (z.B. sinv-56011.edu.hsr.ch) und wählen Sie dort *Jenkins* aus.

3) Klicken Sie im Hauptmenü auf *Neuen Job anlegen*.

Abbildung 145: Anleitung Jenkins: Hauptmenü

4) Geben Sie Ihrem Job einen Namen, wählen Sie *„Free Style"-Softwareprojekt bauen* aus und klicken Sie auf *OK*.

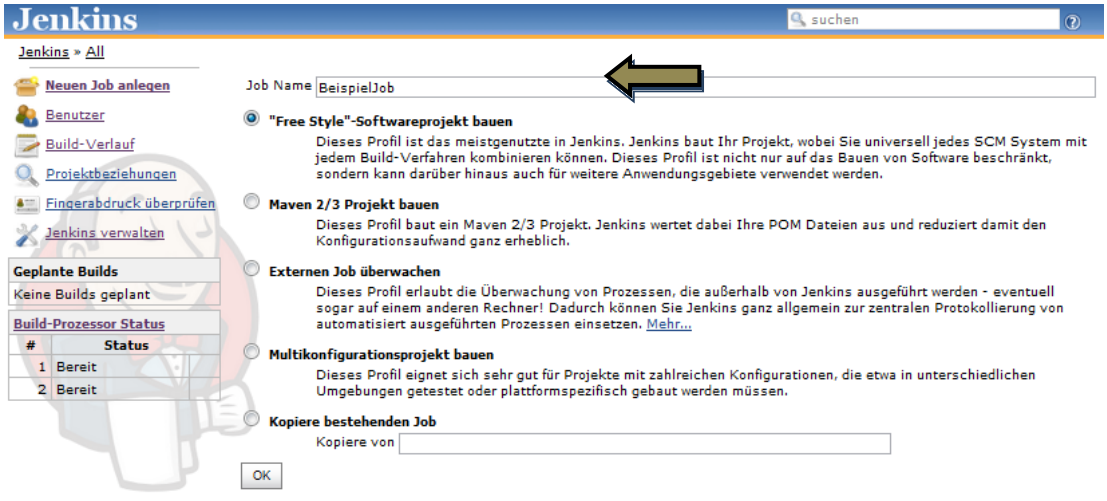


Abbildung 146: Anleitung Jenkins: Neuer Job

5) Konfiguration für SVN:

Den Pfad zum Repository eintragen.

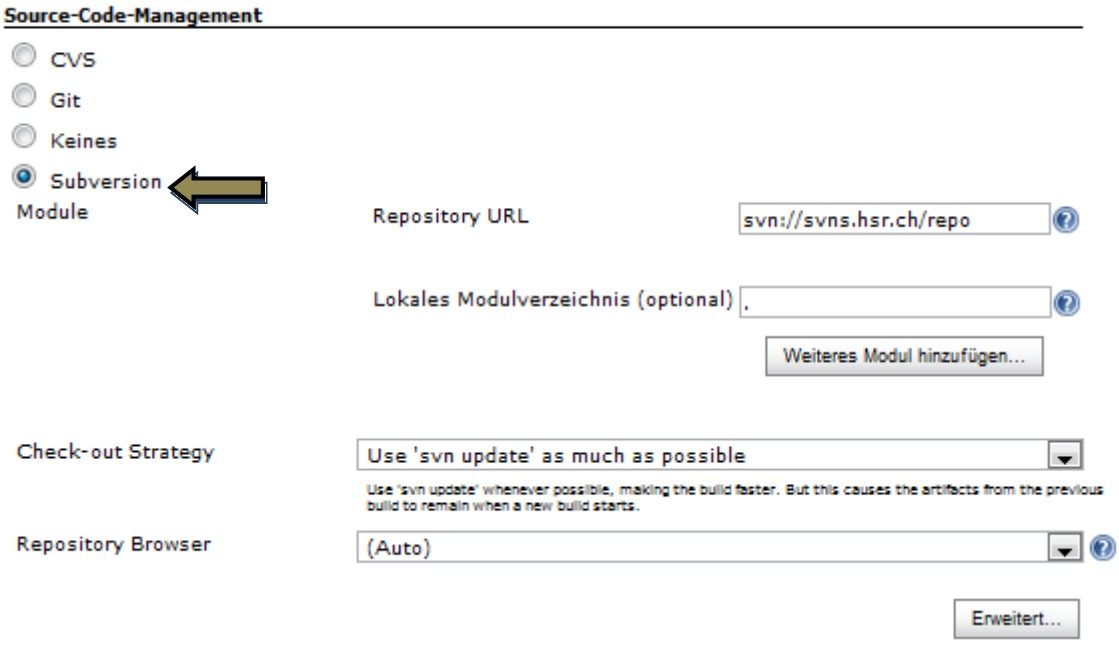


Abbildung 147: Anleitung Jenkins: Integration SVN Repository

6)

Konfiguration für Git:

Den Pfad zum Repository eintragen.

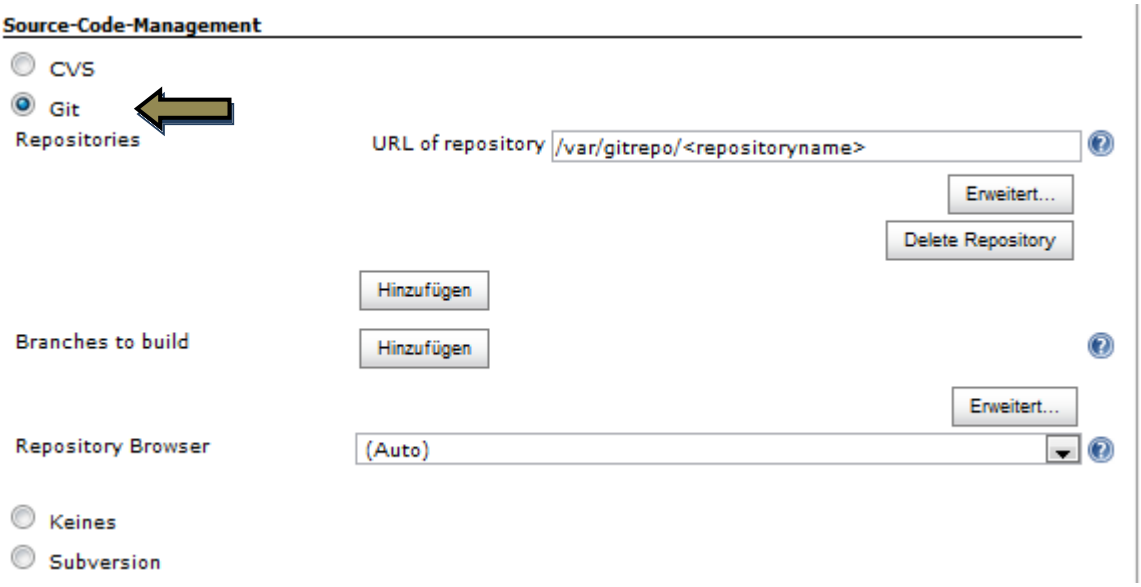


Abbildung 148: Anleitung Jenkins: Integration Git Repository

7) Hier wird konfiguriert wann ein Build gestartet werden soll.
Mit der Eingabe von * * * * * befehlen Sie dem Build Server nach jeder Änderung im ausgewählten Repository einen neuen Build zu erstellen.

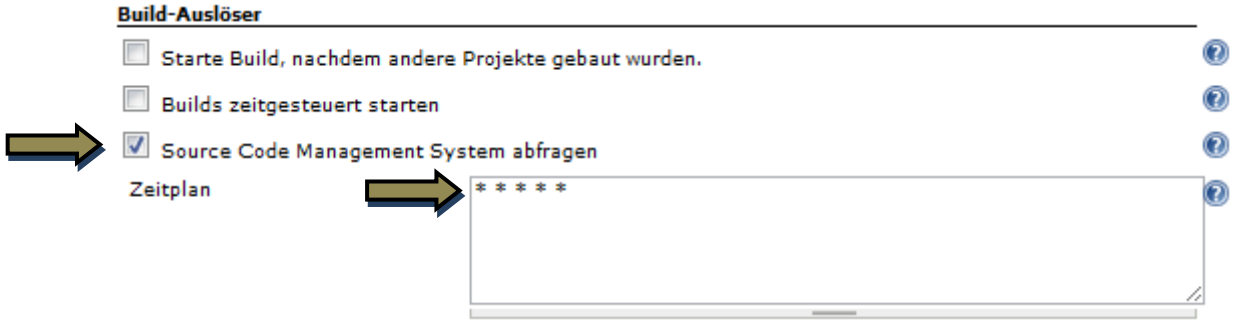


Abbildung 149: Anleitung Jenkins: Build Auslöser

8) Als Buildverfahren werden Sie wohl Ant verwenden:
- Target leer bedeutet default Target wird aufgerufen
- Ant Build Datei leer bedeutet build.xml wird im Stammverzeichniss gesucht

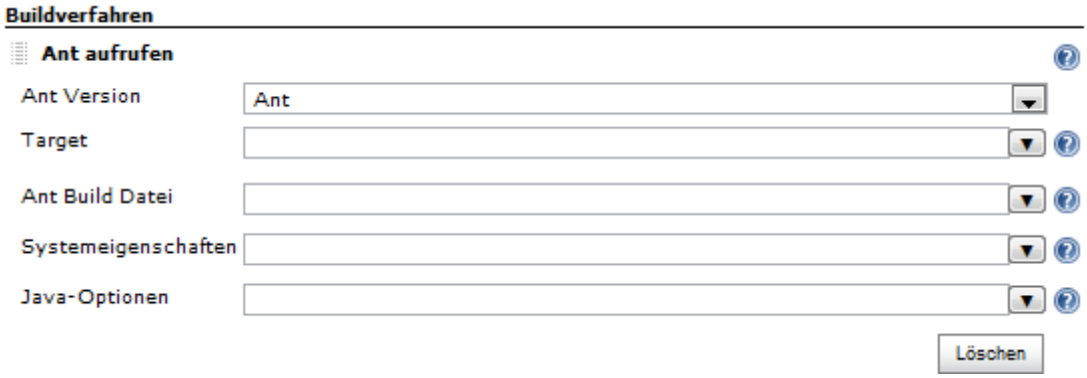


Abbildung 150: Anleitung Jenkins: Buildverfahren

9) Klicken sie am nun zu unterst auf *Übernehmen*

10) Bei den Post-Build-Aktionen können die Auswertungen aktiviert werden. Hierfür muss jedoch zuerst das Ant build.xml erstell werden. Falls Sie dazu Hilfe benötigen, sehen Sie sich das build.xml des Beispielprojektes an unter Punkt *Beispiel Ant - File*.

11) Als erstes sollt nun ein Build-durchgeführt werden. Folgende Angaben müssen gemacht werden:

a. FindBugs Ergebnisse: Pfad zum FindBugs Report
b. FindBugs Zusammenfassung
c. Veröffentliche JUnti-Testergebnisse: Pfad zu den JUnit reports
d. Veröffentliche Emma Testabdeckung: Pfad zum Emma report

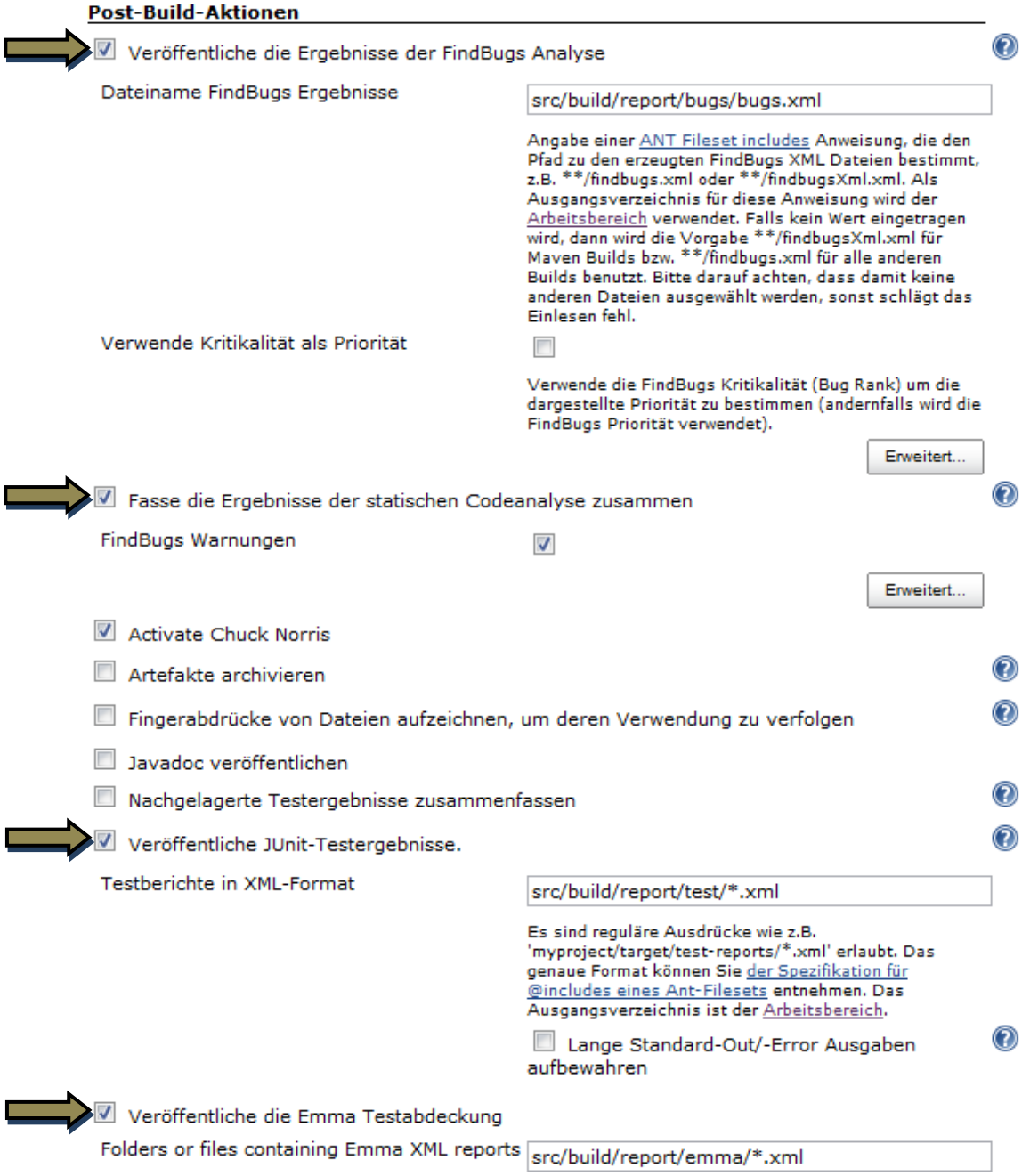


Abbildung 151: Anleitung Jenkins: Post-Build-Aktionen

8.7.3 Anwendungstipps

8.7.3.1 Beispiel Ant - File

Ein Beispiel eines Ant-Build-File finden Sie im Source Code des Beispielprojektes (direkt im Hauptverzeichnis im build.xml).

Weitere Hilfe bietet: http://ant.apache.org/manual/index.html

8.7.3.2 Build starten

Mit der im Setup empfohlenen Einstellung löst jede Änderung im Source Code Management System ein Build-Prozess aus.

Dies kann auch beliebig, durch ein Klick auf *Jetzt bauen* manuell gestartet:

Abbildung 152: Anleitung Jenkins: Jobmenü 1

8.7.3.3 Zugriff auf Build Resultate

Um auf die Resultate eines Build-Prozesses zuzugreifen gehen Sie wie folgt vor:

Klicken Sie auf *Arbeitsbereich*.

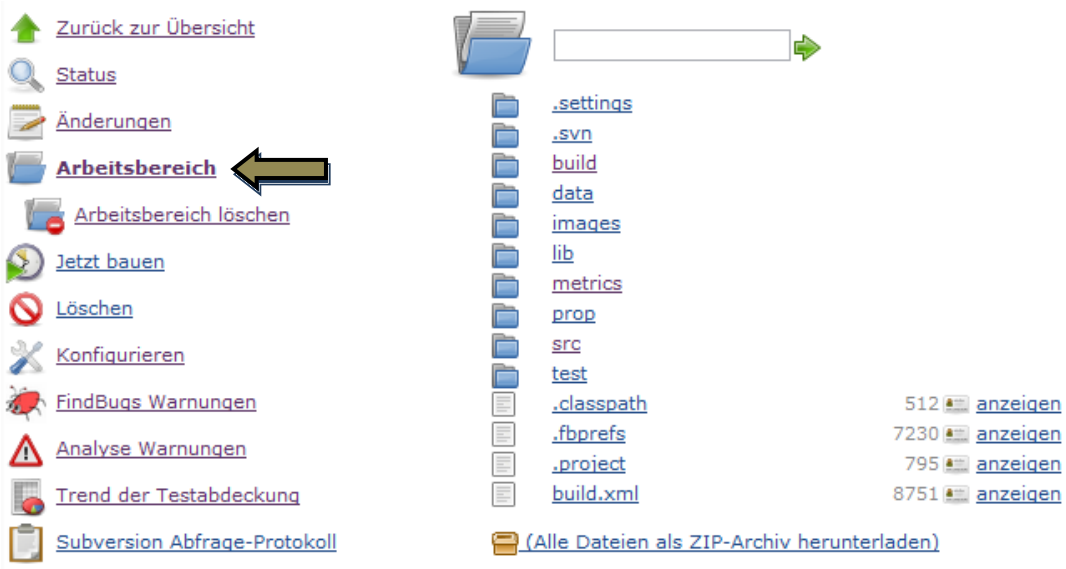


Abbildung 153: Anleitung Jenkins: Arbeitsbereich

8.7.3.4 Zugriff auf weitere Resultate

Reports von FindBugs und der Testabdeckung (Emma) finden Sie unter den folgenden Punkten.

Abbildung 154: Anleitung Jenkins: Jobmenü 2

Anhang

9. Glossar

9.1 Begriffe

Begriff	Beschreibung
State oft the Art	Bezeichnet der aktuell höchst verfügbare Entwicklungsstand der Technik
Trac	Freies, webbasiertes Projektmanagement-Tool zur Software Entwicklung
Open Source	Software, deren Quelltext öffentlich zugänglich ist
Metrik	Eine quantitative Skala und eine Methode, mit welcher der Wert ermittelt werden kann, den ein Indikator für ein bestimmtes Softwareprodukt aufweist.
Redmine	Open Source Projektmanagementsoftware
Fedora	Linux-Variante von Red Hat
Ubuntu	Linux-Variante der Ubuntu Foundation
Ruby	Eine Interpretiert und Objektorientierte Programmiersprache
Rails	In Ruby geschriebenes offen Web Application Framework
Rake	Ruby make wird zur Übersetzung von Ruby benötigt
PostgreSQL	objektrelationales Datenbankmanagementsystem
MySQL	relationales Datenbankverwaltungssystem
Apache Server	ein gängiger WebServer
One-Click-Installer	ein Installations Wizard
BitNami	Firma, die Support und Hosting von Open Source Produkten liefert
Cloud Service	Rechenleistung sowie Speicherplatz im Netzwerk anbieten
Python	Eine Interpretiert und Objektorientierte Programmiersprache
Eclipse	Eclipse ist ein kostenfreies Programmierwerkzeug zur Entwicklung von Software verschiedenster Art.
NetBeans	NetBeans ist eine Entwicklungsumgebung, die komplett in der Programmier-sprache Java entwickelt wurde und auf der NetBeans Plattform läuft.
Structure101	Visuelle Repräsentation von Informationen über Software Systeme basierend auf deren Struktur, Grösse, Geschichte oder Verhalten.
Plug-In	Ist ein Computerprogramm, das in ein anderes Softwareprodukt „eingeklinkt“ wird und damit dessen Funktionalität erweitert.
Package	Pakete (Ordnerstrukturelemente), die bei der Strukturierung von Code helfen.
PuTTY	freies Secure Shell-, Telnet- und Rlogin-Programm
STAN	Tool zurStruktur Analyse
Wizard	Eine schrittweise geführte Instrunktionsanweisung.

Tabelle 9: Glossar: Begriffe

9.2 Abkürzungen

Abkürzung	Beschreibung
SEPEnv	Software Engineering Project Environment (Softwareentwicklungsumgebung für SE Projekte)
bbodenma	Beat Bodenmann
czurbuch	Christian Zurbuchen
ECTS	European Credit Transfer and Accumulation System
PM	Projekt Management
SE	Software Engineering
HTML	Hypertext Markup Language; Textbasierte Sprache, welche zur Strukturierung von Inhalten (Texte, Bilder, Links) dient; Darstellung im Webbrowser
RUP	Rational Unified Process; Vorgehensmodell zur Software Entwicklung
HSR	Hochschule für Technik Rapperswil
SVN	Apache Subversion; Freie Software zur Versionsverwaltung von Dateien und Verzeichnissen
SAD	Software Architecture Document; beinhaltet Design und Architektur des Produktes
XML	Extensible Markup Language; ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdaten.
CSV	Comma Separated Values; Das Dateiformat CSV beschreibt den Aufbau einer Textdatei zur Speicherung oder zum Austausch einfach strukturierter Daten.
C#	Programmiersprache die auf dem .Net Framework aufbaut
bll	Business Logic Layer
dal	Data Access Layer
CRUD	Create, Read, Update, Delete

Tabelle 10: Glossar: Abkürzungen

10. Verzeichnisse

10.1 Literatur

- UML 2 und Patterns angewendet, Craig Larman, ISBN 978-3-8266-1453-8
- Eclipse/NetBeans (http://www.retokiefer.com/archives/ide-shootout-2011-eclipse-vs-netbeans-vs-intellij-idea/) (Stand 30.09.12)
- Eclipse/NetBeans (http://www.devx.com/Java/Article/34009/0/page/1) (Stand 30.09.12)
- Vorlesungsunterlagen Modul Software Engineering 2 Stand FS2011
- Eclipse Metrics (http://www.stateofflow.com/projects/16/eclipsemetrics)
- Eclipse Metrics Plugin (http://metrics.sourceforge.net/)
- STAN (http://stan4j.com/)
- Structure101 (http://www.headwaysoftware.com/#intro)
- JDepend (http://clarkware.com/software/JDepend.html
- Artikel mit Vergleich von GUI Builder für Eclipse (Stand 18.10.2011) (http://wiki.computerwoche.de/doku.php/programmierung/gui-builder_fuer_eclipse)
- WindowBuilder Pro (http://code.google.com/intl/de-DE/javadevtools/wbpro/index.html)
- Eclipse Marketplace (http://marketplace.eclipse.org/)
- FindBugs (http://findbugs.sourceforge.net/)
- EclEmma (http://www.eclemma.org/userdoc/index.html)
- Checkstyle (http://checkstyle.sourceforge.net/index.html)
- SVN (http://subversion.apache.org/)
- Git (http://git-scm.com/)
- Redmine (http://www.redmine.org/)
- Trac (http://trac.edgewall.org/)
- Jenkins (http://jenkins-ci.org/)
- Emma(http://emma.sourceforge.net/userguide/userguide.html)
- Ant (http://ant.apache.org/manual/index.html)
- Jenkins (http://pkg.jenkins-ci.org/debian/)
- SSH-Server (http://wiki.debian.org/SSLkeys)

10.2 Zitate

10.3 Tabellen

10.4 Abbildungen