Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Till Streckwaldt

Continuous Integration:

Webbasierte agile Softwareentwicklung in interdisziplinären Projekten

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Till Streckwaldt

Continuous Integration:

Webbasierte agile Softwareentwicklung in interdisziplinären Projekten

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Birgit Wendholt
Zweitgutachter: Lutz Behnke MSc

Eingereicht am: 12. Dezember 2013

Till Streckwaldt

Thema der Arbeit
Continuous Integration: Webbasierte agile Softwareentwicklung in interdisziplinären Projekten

Stichworte
Continuous Integration, Continuous Integration software, Agile Developement, Jenkins, Maven, Ant, Build-Management, Versionierung, Webapplikation, Projektverwaltung, Informatik

Kurzzusammenfassung
Um künftige Kooperationen und Projekte vom Department Informatik und informatikfernen Departments der HAW zu erleichtern, soll Agile Softwareentwicklung eingeführt werden. Umgesetzt wird dies mit dem Paradigma "Continuous Integration", also das stetige Einfügen jeder Änderung in das Hauptprojekt und das Testen einer jeden Änderung. Für einen vereinfachten Einstieg in diese Art der Entwicklung, werden die Schritte durchgeführt, einen Versionierungs- und Buildserver zu integrieren und mit einer benutzerfreundlichen Webanwendung zu bedienen.

Till Streckwaldt

Title of the paper
Contiuous Integration: web-based agile software development interdisciplinary projekts

Keywords
Continuous Integration, Continuous Integration software, Agile Development, Jenkins, Maven, Ant, Build-Management, revision-control, web-application, project administration, Computer Science

Abstract
To ease future cooperation of the Computer Science Department and other Deparments, unrelated to computer science, of the HAW, Agile Software Development is to be introduced. This will be implemented with the paradigm ”Continuous Integration”, which means the continued integration of each and every change into the main line and testing of every change. To simplify the entry into these means of development, a revision control and build server will be set up, supported by a user friendly webapplication.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

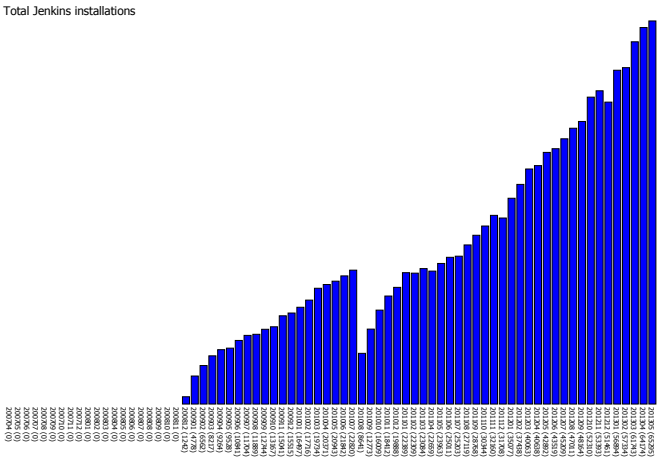
1.1 Motivation

In immer mehr Bereichen wachsen unterschiedliche Disziplinen fortwährend stärker zusammen und sind auf gegenseitige Abstimmung bei der zielgerichteten Entwicklung von Software angewiesen. Designer für Usability und User-Interfaces müssen sich mit Programmieren abstimmen oder Produktdesigner ihre Ideen umsetzen lassen. Heute und in Zukunft ist es wichtig, dass die Softwareentwicklung agil und teamorientiert erfolgt. Dazu wurden in den letzten Jahren neue Systeme entwickelt und Software auf dem Markt gebracht um dieses Vorhaben zu unterstützen. Zum einen agile Methoden, wie Paarprogrammierung und testgetriebene Entwicklung, zum anderen agile Prozesse, wie Scrum oder Kanban. Diese Vorgehensweisen führen dann zu unterstützenden Systemen wie Continuous Integration[1].

In bestimmten Kursen an der Hochschule für angewandte Wissenschaften Hamburg wird den Studenten der Informatik bereits eine agile Methode näher gebracht, es ist aber von Vorteil, wenn die Studierenden so früh wie möglich auch in die Arbeitsweise der Continuous Integration eingeführt werden. Durch eine besonders auf den Bedarf der HAW abgestimmte Software- und Serverlösung soll diese agile Methode in die Hochschulpraxis integriert werden. Ein ganz besonderer Anwendungsbereich sind Projekte, die in Abstimmung und Zusammenarbeit zwischen dem Department Informatik und der Informatik fernen Departments erfolgen. Dies sind beispielsweise Installationen mit Licht- und Farbspielen im Rahmen von Computational Spaces. Dabei wechseln die Spezifikationen verhältnismäßig häufig und es müssen stetig aktuelle Versionen des Programms an andere Teams ausgeliefert werden.

Ein System dieser Art lässt sich Anhand der Zielgruppen weiter festlegen, so gibt es zum einen Studenten in höheren Semestern, die bereits Erfahrungen in unterschiedlichen Disziplinen gesammelt haben und zusammen mit Kommilitonen an Projekten arbeiten und unterschiedliche Spezifikationen umsetzen. Die zweite Gruppe sind Studenten im ersten Semester. Diese sollen so früh wie möglich Kontakt mit agilen Methoden und den Prinzipien von Continuous

[1] im Folgenden auch CI



Quelle: http://stats.jenkins-ci.org/jenkins-stats/svg/svgs.html (05.07.2013)

Abbildung 1.1: Installation der CI-Software Jenkins von 2007-2013

Integration haben. Es erfolgt eine langsame Heranführung an diese Art zu Programmieren im Umfeld einfacher Aufgaben. Um einen Prototypen umfassend und erfolgreich zu testen, ist es effizienter eine Plattform für die höheren Semester zu schaffen. Die Machbarkeit des Austausches in Projekten, die Rahmenbedingungen und prototypische Testläufe können im Anschluss leicht auf weniger komplexe Systeme für andere, früher im Kurrikulum liegende, Veranstaltungen übertragen werden.

Dazu bietet sich natürlich Continuous Integration an, auf der einen Seite arbeiten Studenten an der Software, testen diese und liefern nach jeder Änderung einen aktuellen Stand des Programms aus. Die so generierten Artefakte können dann von anderer Stelle weiter genutzt und integriert werden.

1.2 Zielsetzung

Für die einfache Verwendung in unterschiedlichen Kontexten ist eine bedürfnisgerechte Erstellung der kompletten Umgebung vorgesehen. Mit den nötigen Projektdaten wird eine neue,

konfigurierbare Projektinstanz erstellt, deren Funktionsumfang dann, gemäß den Anforderungen, für die Projektverwaltung und Continuous Integration zuständig ist.

Die Projektinstanzen nutzen unterschiedliche Module und ein Programm, das die Funktionen bündelt und den Endnutzern zur Verfügung stellt. Das Ziel ist es folgende Schritte bei Softwareprojekten abzubilden:

1. Versionsverwaltung von Sourcen
2. Automatischer Build der Sourcen
3. Verwaltung externer Abhängigkeiten
4. Automatische Tests und Darstellung der Testergebnisse
5. Analyse anhand von Code-Metriken
6. Bereitstellung der Artefakte

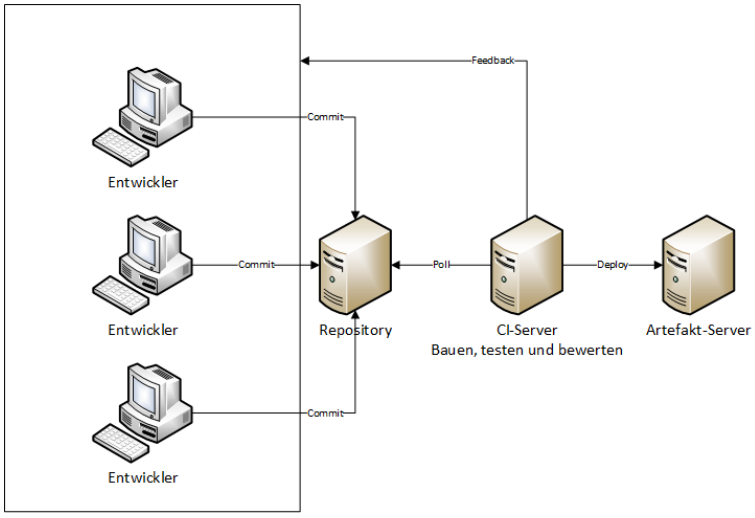


Abbildung 1.2: CI-Umgebung

Diese Schritte werden im allgemeinen auch als die Prinzipien von Continuous Integration bezeichnet und stellen den Mindestumfang für eine voll integrierte Umgebung dar.
Damit die Instanzen in der Lehre in niedrigen Semester und in interdisziplinären Projekten eingesetzt werden können, ist ein angepasstes Rollenkonzept nötig. Aus den Anforderungen

ergeben sich folgende Elemente. Die Reihenfolge entspricht dabei weniger umfangreichen Rechten.

1. *Professoren* haben umfassende Rechte Projekte zu erstellen und zu modifizieren, Benutzer anzulegen, Rechte zu vergeben und auch Zugriff auf alle Funktionen der unteren Gruppen
2. *Student-IT* kann, zusammen mit anderen, an einem Projekt teilnehmen und dieses verwalten, Gruppen bilden, hat Zugriff auf Tests und Codemetriken
3. *Student* kann passiv an einem Projekt teilnehmen und Artefakte herunterladen, hat Zugriff auf Testergebnisse und kann Gruppen bilden

Besonderer Wert wird auf die Erweiterbarkeit des Serverpaketes gelegt, da sich in der Softwareentwicklung die Schwerpunkte schnell verschieben können und neue Programmiersprachen und Umgebungen berücksichtigt werden müssen. Artefakte müssen dann für neue Zielplattformen gebaut und bereitgestellt werden. Neu hinzugekommene Programmiersprachen müssen integriert und anschließend den selben, gewohnten Funktionsumfang bieten. Die Wartbarkeit spielt zwar auch sehr wichtige Rolle, ist aber, da durch gewählte Module der Funktionsumfang auch für die Zukunft sichergestellt wird, nicht Teil dieser Arbeit.

1.3 Gliederung

Das zweite Kapitel gibt eine Einführung in den klassischem Funktionsumfang einer CI-Plattform. Da Continuous Integration in der Anwendung relativ neu ist, werden an dieser Stelle die Prinzipien von CI, mit allen Einzelschritte erklärt. Eine Plattform unterstützt den Anwender zwar bei der Einhaltung durch stetige Rückmeldungen, einen wichtigeren Teil macht aber die Einhaltung von Grundregeln aus. Abschließend wird auf vergleichbare Arbeiten eingegangen.

Das dritte Kapitel ist der Problemanalyse im Gesamten gewidmet, die funktionalen und nichtfunktionalen Anforderungen werden definiert. Auf die Zielgruppen und deren Rollen, sowie Anwendungsfälle wird eingegangen. In der Zusammenfassung werden alle Aspekte gebündelt betrachtet und mögliche Risiken abgegrenzt. Um die Anforderungen dann zu erfüllen werden die Kriterien definiert, anhand derer man dann eine Software und Module auswählt.

Das vierte Kapitel befasst sich mit der Auswahl einer der Lösung zugrunde liegenden Software. Da im Bereich CI eine komplette Neuentwicklung keinen Sinn macht, wird eine CI-Plattform

mit konfigurierbaren und modifizierbaren Paketen vorgestellt. Der Auswahlprozess wird anhand einiger Beispiele im Detail diskutiert.

Kapitel Fünf beschreibt die konkrete Integration und Konfiguration der Serverumgebung anhand der ausgewählten Pakete. Das erarbeitete Rollenkonzept wird umgesetzt und in die fertige Plattform integriert. Wichtiges Merkmal dabei sind unterschiedlich komplexe Ansichten, die einen angemessenen Informationsgehalt aber keine zu hohe Komplexität bieten. Ein wichtiger Punkt an der Hochschule, aber auch in anderen Umgebunden ist der Datenschutz. Wie sich Benutzer im fertigen System authentifizieren und autorisieren wird diskutiert und eine Lösung präsentiert.

Im letzten Kapitel werden die Arbeit zusammengefasst und mögliche Verbesserungen und Erweiterungen der Arbeit beschrieben.

2 Grundlagen

In diesem Kapitel findet eine Einführung in die Softwareentwicklung statt. Im Fokus liegen die agilen Methoden und CI. Es wird erläutert, warum Continuous (Code) Integration sich damit in Einklang bringen lässt und sogar große Vorteile bringt. Im letzten Teil konzentriert sich dieses Kapitel dann auf die CI und erläutert alle Schritte, die notwendig sind, um eine vernünftige Methodik damit zu erreichen. Abschließend werden kurz vergleichbare Arbeiten beschrieben.

2.1 Agile Softwareentwicklung

Agile Softwareentwicklung heißt, eine inkrementelle und iterative Entwicklung auf Basis von Lösungen, die im Zusammenspiel mehrerer Teams und mit dem Kunden entwickelt werden. Flexible Reaktionen auf sich ändernde Bedingungen gehören ebenso dazu, wie der stetige Diskurs unter- und miteinander. Das Agile Manifest [David Cohen (2003)] besagt in den vier Kernaussagen :

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Die schon seit einiger Zeit bestehende, aber noch lange nicht abgeschlossene Geschichte der Agilen Softwareentwicklung, ist gut im Report von Cohen, Lindvall und Costa beschrieben: “Agile Software Development”. [David Cohen (2003)] Eine sehr ausführliche Beschreibung der Agilen Softwareentwicklung stammt von einem der Pioniere auf diesem Gebiet und einem der Urheber des Agilen Manifests, Alistair Cockburn. [Cockburn (2000)]

Der verschobene Fokus auf die Interaktion zwischen den an einem Projekt beteiligten Personen mag auf den ersten Blick im Widerspruch mit der strikten Umsetzung eines Prozesses wie Continuous Integration stehen, wenn man aber einige der zwölf Prinzipien, auf denen

das Agile Manifest aufgebaut ist, zu Rate zieht, wird deutlich, wie nützlich ein Prozess wie CI ist.

“1. Customer Satisfaction by rapid delivery of useful software” - bei der CI erhält der Kunde durch fortwährende Tests und anschießender Code-Integration und - Verteilung immer aktuelle, lauffähige Ergebnisse.

“3. Working software is delivered frequently (weeks rather than months)” - durch kontinuierliche Integration jeder Änderung in die Hauptlinie wird sogar tägliche Software-Auslieferung erreicht.

“9. Continuous attention to technical excellence and good design” - durch die Überwachung von Softwaremetriken, Einhaltung von Code-Richtlinien und Tests wird eine überdurchschnittliche Code-Qualität und technische Fehlerfreiheit abgesichert.

Somit dient CI dazu, die bereits bestehende Agilität eines Projektes weiter zu verbessern und zu erhalten.

Wie sich Continuous Integration in die Agile Softwareentwicklung einbringen lässt und welche Vorteile es birgt, damit haben sich bereits die Autoren Martin R. Bakal, Jennifer Althouse und Paridhi Verma von IBM befasst. Ihre Analyse, kommt zum selben Schluss wie diese Arbeit [Martin R. Bakal (2012)]

Das Ziel der Anwendung dieser Prinzipien in einem Hochschulkontext und spezieller in interdisziplinären Projekten, ist genau die unkompliziertere Zusammenarbeit unterschiedlich ausgerichteter Teams beziehungsweise Personen, also von Kunden, Abnehmern und Entwicklern. CI ist dabei das perfekte Werkzeug um die Einhaltung agiler Methodik zu garantieren und den Studenten und Lehrbeauftragen unter die Arme zu greifen.

2.2 Continuous Integration

Ein Problem, welches besonders große Teams, aber auch kleine Teams, von Softwareentwicklern haben, ist, wie man die einzelnen Ergebnisse von Individuen zusammenbringen kann. Es ist ein leichtes zu sagen, nach jeder Codeänderung muss diese für alle zugänglich gemacht werden, dieses auch zu tun ist etwas gänzlich anderes. Weitere Probleme entstehen durch verschleppte

Integration von Änderungen, deren Beseitigung im weiteren Verlauf, mit wachsender Anzahl, immer mehr Zeit in Anspruch nimmt. [siehe Paul M. Duvall (2007) und Fowler (2006)]

Aus dieser Problematik und einer Gemeinschaft von Programmierern aus dem Umfeld des Extreme Programming[1] ist der Gedanke der Continuous Integration erwachsen, der diese Praktiken auf die Spitze treibt und jeden Schritt weitestgehend automatisiert und immer wieder ablaufen lässt. Aus dem Bereich des Extreme Programming kommt Martin Fowler[2], der mit einer kurzen Abhandlung zum Thema CI eine oft genutzte Ressource geschaffen hat, die einen schnellen Überblick gewährt [Fowler (2006)]. Die Herkunft von CI macht auch eine Erwähnung des Begriffs im Standardwerk zu Extreme Programming [Beck (2004)] von Kent Beck deutlich.

Im Folgenden gibt diese Arbeit nun einen Überblick über die einzelnen Schritte der Continuous Integration, welche Probleme damit jeweils zu lösen sind und welche Ansätze es in dem Bereich gibt, diesen Schritt umzusetzen. Die einzelnen Schritte sollen dabei auf einer dedizierten Maschine ablaufen, welche die Verwaltung des Projektes übernimmt. Dabei berufe ich mich größtenteils auf die Arbeiten von Martin Fowler [Fowler (2006)] und Paul M. Duvall [Paul M. Duvall (2007)]. Wurden weitere Quellen genutzt, sind diese getrennt angegeben.

2.2.1 Versionskontrolle

In der Versionskontrolle (auch Software-Repository oder Quellcode-Verwaltung) werden alle für ein Projekt relevanten Dateien abgelegt und zentral verwaltet. Unabhängig von Continuous Integration ist es mittlerweile der Standard geworden, um mehreren Entwicklern die gemeinsame Arbeit zu ermöglichen. Zu Beachten ist, dass nicht nur der Quellcode dort verwaltet wird, sondern alle Dateien die für ein erfolgreichen Bauvorgang nötig sind. Dazu gehören natürlich der Quellcode, aber auch die Testfälle, Properties-Dateien, Bibliotheken und auch Dokumentationen. Martin Fowler fasst es mit den Worten zusammen, dass man auf einer völlig neuen Maschine, nach einen Checkout aus der Versionierung das komplette Projekt bauen können muss. [Fowler (2006)] (Einschränkungen sind natürlich gegeben, wie zum Beispiel das Vorhandensein des Java Development Kits)

Weitere Funktionen der Versionierung sind die Möglichkeit, Zweige von einer Software anzulegen und andere Entwicklungen abzubilden. Der Hauptentwicklungszweig wird dann

[1] Veröffentlichen der Software in kurzen Entwicklungszyklen
[2] http://martinfowler.com/

gängigerweise als “Mainline” bezeichnet.

Noch im Einsatz befindet sind CVS[3], wird aber mehr und mehr von moderneren Systemen mit mehr Einsatzmöglichkeiten und Vereinfachungen abgelöst. Der de facto Nachfolger ist Subversion[4], der Wechsel von CVS kann mit nur geringen Aufwand stattfinden. Eine weitere moderne Lösung mit hoher Verbreitung ist Git[5].

Allen Systemen gleich ist die grundlegende Arbeitsweise:

- Man holt sich den aktuellen Stand des Projektes aus der Versionierung (auschecken; check-out)
- Man nimmt lokale Änderungen vor
- Man lädt die Änderungen in die Versionierung (einchecken; check-in; commit)

Was die Systeme unterscheidet, ist wie Änderungen festgehalten werden, der Umgang mit auftretenden Konflikten und der Ort, an dem die verwalteten Dateien abgelegt sind.

2.2.2 Erstellungsprozess

Nachdem nun an zentraler Stelle eine aktuelle Version des Quellcodes vorgehalten wird, folgt daraus, dass man dort auch die Ergebnisse festhält. Dazu ist ein automatisches Übersetzen des Programms nötig. Wird dieser Prozess einmalig festgehalten, werden Fehler die bei den einzelnen Schritten entstehen können minimiert und jeder beteiligte Entwickler kann sich, bei Erfolg des automatischen Bauens, sicher sein, dass seine Änderungen keine kritischen Fehler (die das Bauen verhindern) verursachen. Das Zitat von Martin Fowler wird in diesem Zusammenhang erweitert, man muss auf einer völlig neuen Maschine, nach einem Checkout aus dem Repository, mit Hilfe eines einzelnes Befehls, ein laufendes System haben. [Fowler (2006)]

Natürlich sind beim Übersetzen eine ganze Reihe von Dingen zu beachten, bevor es autonom passieren kann. Bei sehr großen Projekten, bei denen dieser Vorgang eine lange Zeit in Anspruch nimmt, muss der Zeitpunkt für einen kompletten Build wohl gewählt werden. Alternativ kann man das große Gesamtpaket in logische Einheiten unterteilen, die bei Änderungen

[3] Concurrent Versions System: http://savannah.nongnu.org/projects/cvs
[4] http://subversion.apache.org/
[5] http://git-scm.com/

an dieser Komponente gebaut werden, andere werden dabei nicht angetastet. Eine weitere Frage, die man sich stellen muss, ist, ob man zusätzlich zum regulären Bauvorgang auch Tests durchführen möchte, um die Integrität der Änderungen zu wahren.

Für alle diese Fragen gibt es seit geraumer Zeit eine ganze Reihe von Tools, die diese Arbeiten erledigen. Für die Programmiersprache Java sind dies Ant[6] und das neuere Maven[7]. Bei .NET ist es Nant[8] oder MSBuild[9], um nur einige Beispiele zu nennen.

2.2.3 Testprozess

Wie bereits erwähnt, folgt auf das automatische Bauen, das automatische Testen der Änderungen am Quellcode. Durch den automatisches Bauprozess wird sichergestellt, dass ein Programm überhaupt lauffähig ist (also ohne Fehler gebaut werden kann), nicht aber, ob die Ergebnisse im Anschluss korrekt sind. Durch Testfälle, die eine hohe Codeabdeckung[10] haben, kann genau dieses sichergestellt werden. Allgemein angewandt wird dieser Prozess auch bei der testgetriebenen Entwicklung[11] und dem Extreme Programming.

Es gibt aber einige Dinge zu beachten, die von automatischen Tests erfüllt werden müssen. Zuallererst muss wieder ein einziger Befehl ausreichen, um die Tests ablaufen zu lassen. Nach der Durchführung müssen die Ergebnisse ebenso automatisch ausgewertet werden. Das bedeutet, dass der Fehlschlag eines einzigen Tests dafür sorgt, dass der gesamte Testprozess und als Resultat davon auch der Erstellungsprozess fehlschlägt und dies automatisch erkannt wird.

Hohe Popularität haben die sogenannten xUnit-Frameworks erreicht, spezielle Frameworks für konkrete Programmiersprachen um Tests zu schreiben. Bekanntester Vertreter dürfte JUnit[12]für Java sein, weitere sind NUnit[13] für .NET und CppUnit[14] für C++. Weitergehende Umgebungen für besondere Anwendungen sind zum Beispiel Selenium[15] oder FIT (Framework for Integrated

[6] http://ant.apache.org/
[7] http://maven.apache.org/
[8] http://nant.sourceforge.net/
[9] http://msdn.microsoft.com/en-us/library/0k6kkbsd.aspx
[10] Durch die Tests wird nahezu jeder Punkt im Programm erreicht
[11] Schreibe Tests vor dem eigentlichen Code
[12] http://junit.org/
[13] http://www.nunit.org/
[14] http://sourceforge.net/projects/cppunit/
[15] http://www.seleniumhq.org/

Tests)[16], welches wenig Programmiererfahrung benötigt, da die Tests in anderen Dokumenten, ohne Kenntnisse der Programmiersprache geschrieben werden.

2.2.4 Code-Integration

Unter Code-Integration versteht man, dass stetige einchecken der lokalen Änderungen in die Versionierung. Dies muss geschehen, um allen anderen Entwicklern den aktuellen Stand mitzuteilen, da es natürlich mögliche Abhängigkeiten gibt. Dabei ist zu beachten, dass man nicht einfach blind jede Änderung eincheckt, sondern lokal die selben Schritte durchführt, wie auf der automatisierten Integrationsumgebung. Dazu gehört ein erfolgreicher lokaler Build-Vorgang und das bestehen aller dazugehöriger Tests. Sind diese Anforderungen erfüllt, kann problemlos eingecheckt werden.

Besondere Aufmerksamkeit ist diesem Schritt zu widmen, da er in erster Linie darauf basiert, dass die beteiligten Entwickler die passende Mentalität entwickeln. Minimalanforderung sollte ein Check-In pro Tag sein, optimal wäre das bauen, testen und einchecken einer jeden Änderung. Man selber kann nie den kompletten Umfang der eigenen Änderungen absehen und an welchen Stellen es möglicherweise zu Konflikten kommen kann. Verschleppte Integration und Konflikte erhöhen nur den Zeitaufwand, der nötig ist, diese zu beheben.

Häufige Commits ermutigen Entwickler auch dazu, ihre Arbeit in sinnvolle, innerhalb von Stunden zu erledigende, Pakete zu unterteilen. So sind schneller Ergebnisse möglich und die Entwicklung im Team fällt leichter.

2.2.5 Integrations-Umgebung

Zentraler Punkt dieser Arbeit ist der Continuous Code Integration Server. Eine Umgebung, an der alle Schritte zentral durchgeführt werden. Es werden die Hilfsmittel genutzt, die automatisches Bauen und Testen ermöglichen und nach annähernd jedem Commit in die Versionierung durchgeführt. Da die lokale Entwicklungsumgebung sich natürlich unterscheiden kann, was zum Beispiel installierte Bibliotheken angeht, sollte ein Erstellungsprozess erst als wirklich erfolgreich angesehen werden, wenn dieser auf dem CI-Server erfolgreich war.

[16] http://fit.c2.com/

Beispiele für CI-Server sind CruiseControl[17], Microsoft Team Foundation Server[18] und Jenkins[19].

2.2.6 Verteilung

Der letzte sehr wichtige Schritt ist die Verteilung oder auch Deployment. Die Ergebnisse sollen, nach erfolgreichem Erstellungsprozess, automatisch an andere Stellen verschoben werden. Dies beginnt bei Integrationstests die auf eigenen Maschinen mit der passenden Umgebung stattfindet und endet beim Deployment in eine Produktionsumgebung. Hier kann bereits der Kunde oder Abnehmer der Software Zugriff haben und so stetig den Entwicklungsprozess verfolgen. Optimalerweise wird auf dem Deployment-Ziel bereits eine Installation der Pakete vorgenommen, so dass, ohne weitere Schritte zu tätigen, das Programm genutzt werden kann. Aber bereits das Bereitstellen der gebauten und getesteten Artefakte erleichtert die Arbeiten immens.

2.2.7 Weitere wichtige Elemente

Weitere Elemente, die eine Rollen spielen, sind schnelles Bauen beziehungsweise kontrolliertes Bauen. Die Integrationsumgebung überwacht die Resourcen und hat eine Pipeline, die einzelne Build-Vorgänge steuert und zu passenden Zeiten anstößt. Bei besonders großen, langwierigen Prozessen kann man auch eine feste Zeit vorgeben, wann dieser ablaufen soll. Für den normalen Betrieb ist dann sicherzustellen, dass einzelne Teile unabhängig gebaut und getestet werden können und dies in kürzerer Zeit stattfinden kann.

Das Testen auf die Spitze treiben kann man mit einem Test in einer replizierten Produktionsumgebung. Nahezu alle Parameter sind dabei der endgültigen Umgebung angeglichen und wenn Tests hier erfolgreich sind, sollte es nicht mehr zu Problemen kommen.

Ein wichtiger Bereich für die beteiligten Entwickler, aber auch für andere Stakeholder an einem Projekt sind die Berichte, die aus einer CI-Umgebung erwachsen. Erfolgreiche Erstellungsprozesse und Tests generieren Feedback, das für alle beteiligten Personenkreise einzusehen und leicht zu interpretieren ist. Wichtige Code-Metriken, die den Stand in der Versionierung wiedergeben, lassen sich anzeigen. Sollte ein Build oder ein Test fehlschlagen, muss das System unmittelbar die Person, die den Prozess verursacht hat und möglicherweise weitere Stellen benachrichtigen. Im System selber sollte der Status des Projektes deutlich werden.

[17] http://cruisecontrol.sourceforge.net/
[18] http://msdn.microsoft.com/de-de/vstudio/ff637362.aspx
[19] http://jenkins-ci.org/

2.3 Vergleichbare Arbeiten

Zum Thema Continuous Integration gibt es eine Reihe von Arbeiten und davon befasst sich auch ein Teil damit, welche Hürden zu nehmen sind, soll es zu einer Einführung von CI kommen. Das Buch "Continuous Integration Improving Software Quality and Reducing Risks" von Paul M. Duvall [Paul M. Duvall (2007)] wählt einen ähnlichen Ansatz insofern, dass alle nötigen Schritte beschrieben werden, eine CI-Plattform einzurichten und damit zu arbeiten. Das Paper "Implementing Continuous Integration towards Rapid Application Development" [Fazreil Amreen Abdul (2012)] beschreibt die Schwierigkeiten, aber auch die entstehenden Vorteile durch den Einsatz von CI. Besonders für den Hochschulkontext und die Erfahrungen von Studierenden mit einem neuen CI-System interessant ist das Paper "Assessing Undergraduate Experience of Continuous Integration and Test-Driven Development" [Jon Bowyer (2006)]. Das Fazit dabei ist, die sehr hohe Akzeptanz der Studierenden was CI und Test-Driven Development angeht. Die Vorteile und das allgemeine Vorgehen bei CI ist das Thema des Papers "Continuous integration in agile development" [Martin R. Bakal (2012)]. Das häufig zitierte Werk von Martin Fowler, einfach "Continuous Integration" [Fowler (2006)], soll dabei an dieser Stelle nicht unerwähnt bleiben, da ebenso die nötigen Schritte erläutert werden, dabei aber nicht mögliche Probleme außer acht gelassen werden.

Eine ähnliche Bachelorarbeit an der Hochschule für Angewandte Wissenschaften Hamburg aus dem Jahre 2008 [Kluth (2008)], hat es sich zum Ziel gemacht, ein eigenes Open Source Build-Management System und CI umzusetzen. Der Schwerpunkt der Arbeit liegt auf der Auswahl der einzelnen Werkzeuge, die dafür notwendig sind. Im Gegensatz dazu zielt diese Arbeit auf eine webbasierte Lösung ab, unter dem Einsatz eines gegebenen CI-Servers, und legt den Fokus auf einfache Bedienbarkeit. Die einzelnen Werkzeuge sind den Entwicklern dann freigestellt und können durch vielfältige Plugins integriert werden.

3 Problemanalyse

Um den Systemkontext abzugrenzen und zu definieren, bedarf es einer fachlichen Analyse der Anforderungen. Die Elemente des Projektservers werden analysiert und spezifiziert und in funktionale Anforderungen sowie nicht-funktionale Anforderungen unterteilt. Für die Softwarediagramme in diesem und späteren Kapiteln wird sich der Unified Modelling Language (UML) bedient. Damit lassen sich besonders Zusammenhänge, der Schwerpunkt dieser Arbeit, gut darstellen und der Kontext verdeutlichen.

Die Komplexität der Analyse wird dabei so gering wie möglich gehalten, da es unabdingbar ist, alles vollständig und widerspruchsfrei zu definieren. Erste Ansätze werden so wenig technisch wie möglich gehalten, um sich nicht frühzeitig auf eine Realisierung festzulegen. Konkrete Umsetzungen für die auftretenden Erfordernisse folgen in späteren Kapiteln. Dann wird auf Basis der erarbeiteten Konzepte das technische Design entwickelt und in einen Projektserver integriert.

Sämtliche dargestellten Ideen wurden gemeinsam mit Frau Prof. Dr. Birgit Wendholt zu Papier gebracht, als Hauptabnehmer der Software dienen ihre Vorstellungen als fundamentaler Funktionsumfang der Applikation.

3.1 Systemidee

Um zu einem funktionierenden System zu gelangen, sind diverse Schritte durchzuführen und Details zu beachten. Um die funktionalen und nicht-funktionalen Anforderungen zu beschreiben und einzugrenzen wurde zuallererst eine grundlegende Systemidee ausgearbeitet.

Es sollen unterschiedlich versierte Personen an einem gemeinsamen System teilhaben, dass die Arbeit miteinander an umfangreicheren Softwareprojekten erleichtert. Es kommen unterschiedliche Programmiersprachen zum Einsatz und es gibt verschiedene Zielsysteme, für die, aus einem Softwareprojekt entstehenden, Artefakte. Um eine hohe Akzeptanz sicherzustellen, soll das System möglichst einfach zu bedienen sein, so dass auch Studenten im ersten Semester einen direkten Zugang haben, ohne abgeschreckt zu sein. Besonderen Wert wird auf die stetige

Auslieferung der Artefakte gelegt, damit alle beteiligten Personen stetig am Projekt teilhaben und daran arbeiten können. Für all diese Punkte bietet sich eine webbasierte Continuous Integration Plattform grade zu an, da diese alle Anforderungen abdeckt.

3.2 Funktionale Anforderungen

Die funktionalen Anforderungen eines Softwaresystems grenzen den zu bereitstellenden Funktionsumfang ab, stellen diesen im Kontext zu Nachbarsystemen dar und geben Auskunft über die beteiligen Personengruppen und Akteure. Im ersten Abschnitt wird darauf eingegangen, welche Programmiersprachen und Zielsysteme zu unterstützen sind. Weiter wie die Verwaltung der umfangreichen Bibliotheken und Abhängigkeiten im System geregelt wird. Der zweite Teil konzentriert sich auf die Interessensgruppen im Projektumfeld und die Akteure mit denen interagiert wird. Mehrere Anwendungsfalldiagramme und die konkrete Beschreibung der Anwendungsfälle beziehen im Anschluss alle Teile mit ein und setzt diese in den Anwendungskontext.

3.2.1 Unterstütze Sprachen

In Abstimmung mit dem Abnehmer wurden als Funktionsumfang für den Prototypen folgende Programmiersprachen festgelegt: Java, C und C++ sowie C#. Dies bedeutetet für die CI-Plattform, dass Projekte in diesem Programmiersprachen im Versionierungssystem eingecheckt werden können und vom CI-System verarbeitet werden. Dazu gehört das fehlerfreie Bauen und Testen sowie das korrekte Bereitstellen der resultierenden Artefakte. Des weiteren müssen Werkzeuge zur Code- und Testanalyse bereitgestellt werden, sowie Warnungen und Fehlermeldungen ausgeben werden. Weitere Softwaremetriken, wie Codeabdeckung, Einhaltung von Namensrichtlinien oder sonstige Projektstatistiken, sollen ebenfalls für alle Programmiersprachen angezeigt werden.

3.2.2 Ziele und Umgebungen

Die fertigen Artefakte sollen veranstaltungsbezogen abgelegt werden. Dazu kommen mehrere Lösungen in Frage. Zum einen ein eigener Server, auf dem die Artefakte dann im Anschluss bereitgestellt werden, oder ein Verzeichnis auf dem System der CI-Plattform, welches ebenfalls die Ergebnisse der Softwareprojekte vorhält. Es wird im Produktivbetrieb zwar unterschiedliche Zielsysteme geben (vorgesehen sind Windows, MacOS und Android), da aber keine Integrationstests vom System durchzuführen sind, ist es nicht weiter nötig die Funktion sicherzustellen.

Es muss noch vom System her möglich sein, für diese Ziele einen Build-Vorgang durchzuführen und die Resultate dann anschließend an einem Ort abzulegen.

Die Installation der Software und nötigen Softwarepakete muss dann vom jedem Benutzer selber durchgeführt werden.

3.2.3 Verwaltung von Bibliotheken und Abhängigkeiten

Es gibt auch Projekte, die zusätzlich zu den für den Build-Vorgang nötigen Bibliotheken, weitere Abhängigkeiten besitzen. Diese werden auf dem Deployment-Server ebenfalls bereitgestellt und müssen zuvor manuell eingetragen und für das Projekt als relevant markiert werden. Dieser Vorgang kann nicht automatisch stattfinden, da wie bereits erwähnt keine Integrationstests durchgeführt werden. Die lokale Installation kann durch eine, durch das Projektteam beigefügte, Anleitung erleichtert, aber nicht durch das System abgenommen werden.

3.2.4 Beteiligte Benutzergruppen und Akteure

Akteur: Lehrbeauftragter: Diese Gruppierung umfasst die Professoren und wissenschaftlichen Mitarbeiter mit Lehrauftrag, die eine Veranstaltung halten oder ein Praktikum leiten. Ihnen obliegt es, für die Veranstaltung einen Projekt-Server anzufordern und die benötigten Daten bereitzustellen. Im Betrieb legen Lehrbeauftragte die einzelnen Projekte an, kümmern sich um den korrekten Ablauf und pflegen die Liste der Teilnehmer. Einen Einblick in die Ergebnisse der einzelnen Studierenden ist über die Softwaremetriken und die abzuliefernden Artefakte und Abhängigkeiten möglich.

Akteur: Student-IT: Um eine fachliche Abgrenzung zu informatikfernen Departments zu schaffen gibt es eine Gruppe speziell für Studierende aus der IT. Ferner ist es natürlich möglich, jeden Studierenden, der erweiterte Rechte und Möglichkeiten an einem Projekt zu arbeiten benötigt, in diese Gruppe einzuordnen. Im Gegensatz zur Benutzergruppe “Student”, ist es auf dieser Stufe möglich die Sourcen der Projekt-Instanz zu verwalten (einchecken und auschecken), komplexe Testmetriken abzurufen und Projekteinstellungen vorzunehmen. Alle Punkte für die ein fundierteres Wissen von Abläufen aus der CI und der Informatik im Allgemeinen nötig ist.

Akteur: Student: Als einfachste Stufe sind alle anderen Studierenden zu sehen, die zwar Interesse an der Projekt-Instanz und am erfolgreichen Ergebnis haben, aber nicht aktiv programmieren. Die Abstimmung mit anderen Projektteilnehmern findet auf einer anderen Ebene

statt, im Rahmen des CI-Servers ist diese Gruppe vorrangig an den Artefakten und Abhängigkeiten interessiert. Der Zugriff auf einfache Testergebnisse dient dem schnellen Überblick über den Fortschritt.

Akteur: CI-Server: Dieser Akteur ist der CI-Server für eine bestimmte Veranstaltung. Es werden Benutzer verwaltet, beliebig viele Projekte angelegt und eine Verknüpfung zum Versionierungs-Server und Deployment-Server hergestellt. Ein Administrator ist für die Verwaltung des CI-Servers zuständig, der tagtägliche Ablauf erfolgt aber automatisiert und benötigt, nach einmaliger Einrichtung, keine weiteren Eingriffe von außen.

Akteur: Versionierungs-Server: Ebenfalls in der Umgebung auf dem der CI-Server läuft kommt ein Versionierungs-Server zum Einsatz. Der Quellcode der einzelnen Projektinstanzen wird hier verwaltet und kann ein- sowie ausgecheckt werden. Vom CI-Server aus wird eine Verbindung zur Versionierung hergestellt und bei Bedarf kann diese auch ausgetauscht werden.

Akteur: Deployment-Server: Der Deployment-Server ist eine kontinuierliche laufende Umgebung für die in den Projekt-Instanzen entstehenden Artefakte und Abhängigkeiten und ermöglicht den schnellen Download der Dateien für eine lokale Installation.

Weitere beteiligte Akteure: Ein **Administrator** kümmert sich, anhand der durch den Lehrbeauftragten bereitgestellten Spezifikation, um die Einrichtung und den Betrieb des CI-Servers. Langfristig fällt dieser Person auch die Aufgabe der Wartung und Erweiterung der Umgebung zu.

3.2.5 Verwendete Begriffe

Projekt-Server, CI-Server: Teilergebnis dieser Arbeit mit Benutzerverwaltung und Projektverwaltung. Der Projekt-Server stellt die Funktionen der CI-Kette zur Verfügung.

Versionierungs-Server: Verwaltung des Quellcodes der Projekt-Instanzen.

Deployment-Server: Bereitstellung von Softwareartefakten und externen Abhängigkeiten.

CI-Plattform: Der gesamte Softwarekomplex bestehend aus CI-Server, Deployment-Server und Versionierungs-Server.

Veranstaltung: Eine Vorlesung, Praktikum oder Projektveranstaltung, dieser wird eine CI-Plattform zugeordnet.

Projekt: Die einzigartige Definition für eine Aufgabe innerhalb einer Veranstaltung. Von dieser leiten Studenten "ihre" Projekt-Instanz ab.

Projekt-Instanz: Eine eigene Kopie des Projekts an der die Studierenden arbeiten.

(externe) Abhängigkeiten: Sind nicht für den Bauvorgang nötig, aber für die Installation und den Betrieb auf einem System unabdingbar.

Artefakte: Die Ergebnisse aus einer Projekt-Instanz.

Projektgruppe: Mehrere Studierenden die gemeinsam an einer Projekt-Instanz arbeiten.

3.2.6 Anwendungsfälle

Die folgenden Szenarien sind die typischen Anwendungsbeispiele des Prototypen, dabei wird genauer auf die einzelnen Elemente des Anwendungsfalldiagramms eingegangen und die Aktionen der Akteure genauer beschrieben. Erfolgsfall und Ausnahmen werden elaboriert. Die Anwendungsfälle sind dabei in eine Reihenfolge gebracht, die dem Ablauf der Programmnutzung entspricht. Zur besseren Übersicht, welcher Akteur Zugriff auf welche Funktion hat, gehen den Anwendungsfällen mehrere Übersichtsdiagramme voraus.

3.2.6.1 Anwendungsfalldiagramme

In den Diagrammen (Abb. 3.1, Abb. 3.3, Abb. 3.4, Abb. 3.5) wird, aufgeschlüsselt auf die einzelnen Benutzerrollen, eine Übersicht über die Anwendungsfälle gegeben. Im Anschluss erfolgt die konkrete Beschreibung der einzelnen Fälle. Die Diagramme dienen der besseren Übersicht, der doch sehr umfangreichen Use-Cases.

3.2.6.2 Auflistung der Anwendungsfälle

ID und Name	**UC1.1: Neuen Projektserver erstellen**
Beschreibung	Für eine Veranstaltung soll ein Projektserver angelegt werden um die gemeinsame Arbeit und CI zu ermöglichen
Beteiligte Akteure	Akteur: Lehrbeauftragter, Akteur: Administrator und Akteur: CI-Plattform
Auslöser	Neuer CI-Server nötig
Vorbedingungen	Für die Veranstaltung gibt es noch keine CI-Plattform oder es soll ein neuer Server im Auslieferungszustand angelegt werden
Nachbedingungen	Neue CI-Plattform für die Veranstaltung ist angelegt
Ausnahmen	Für eine Veranstaltung in einem Semester soll es nicht mehrere CI-Plattformen geben

Tabelle 3.1: Anwendungsfall “Neuen Projektserver erstellen”

ID und Name	**UC1.2: Benutzer anmelden**
Beschreibung	Der Benutzer meldet sich mit seinen Kontodaten an der CI-Plattform der Veranstaltung an
Beteiligte Akteure	Akteur: Student, Student-IT, Lehrbeauftragter und Akteur: CI-Server
Auslöser	Teilnahme vom Benutzer an einer Veranstaltung
Vorbedingungen	CI-System der Veranstaltung ist aufgesetzt und bereit, Benutzerkonten angelegt und die Rechte vergeben
Nachbedingungen	Benutzer ist angemeldet und kann das System verwenden
Ausnahmen	Wenn das System noch nicht vollständig eingerichtet ist, können sich Benutzer mit niedriger Stufe möglicherweise nicht nicht einloggen

Tabelle 3.2: Anwendungsfall “Benutzer anmelden”

ID und Name	**UC1.3: Studenten hinzufügen**
Beschreibung	Die Studenten, die an der Veranstaltung teilnehmen, sollen zur CI-Plattform hinzugefügt werden, um dann die gemeinsame Arbeit zu ermöglichen
Beteiligte Akteure	Akteur: Lehrbeauftragter und Akteur: CI-Server
Auslöser	Neue Studierende nehmen an einer Veranstaltung teil
Vorbedingungen	CI-Plattform für die Veranstaltung existiert und die Teilnehmerliste liegt vor
Nachbedingungen	Studierende sind in der CI-Plattform angelegt und können sich mit ihren Benutzerdaten anmelden
Ausnahmen	Der Lehrbeauftragte hat die endgültige Kontrolle über die Liste der Studierenden, möglicherweise werden auch weitere Studenten, die nicht auf der Teilnehmerliste sind, hinzugefügt

Tabelle 3.3: Anwendungsfall “Studenten hinzufügen”

ID und Name	**UC1.4: Rechte vergeben und verwalten**
Beschreibung	Den angelegten Studenten werden die passenden Rechte zugeteilt
Beteiligte Akteure	Akteur: Lehrbeauftragter und Akteur: CI-Server
Auslöser	Neue oder geänderte Studenten auf der CI-Plattform
Vorbedingungen	Student ist hinzugefügt
Nachbedingungen	Student hat die passenden, geänderten Rechte
Ausnahmen	

Tabelle 3.4: Anwendungsfall “Rechte vergeben und verwalten”

ID und Name	**UC1.5: Neues Projekt anlegen**
Beschreibung	Im Rahmen einer Veranstaltung gibt es ein neues Projekt, mit bestimmten Randbedingungen, Abhängigkeiten und vorgegeben Sourcen
Beteiligte Akteure	Akteur: Lehrbeauftragter und Akteur: CI-Server
Auslöser	Ein Lehrbeauftragter legt für einen Projektserver ein neues Projekt an
Vorbedingungen	Projektserver ist aufgesetzt
Nachbedingungen	Projekt ist angelegt und optional freigegeben und kann dann von Studentengruppen benutzt werden
Ausnahmen	Projekt kann nur angelegt, aber nicht freigegeben werden, damit ist es nur für den anlegenden Lehrbeauftragten sichtbar

Tabelle 3.5: Anwendungsfall “Neues Projekt anlegen”

ID und Name	**UC1.6: Projekt verwalten**
Beschreibung	Randbedingungen, Abhängigkeiten oder vorgebene Sourcen für ein bestehendes Projekt sollen angepasst werden
Beteiligte Akteure	Akteur: Lehrbeauftragter und Akteur: CI-Server
Auslöser	Anpassen der Projekteinstellungen
Vorbedingungen	Projekt ist angelegt
Nachbedingungen	Für das Projekt und die von diesem Projekt abgeleiteten Projekt-Instanzen werden die geänderten Einstellungen übernommen
Ausnahmen	Hiermit lassen sich die Änderungen in den Projekt-Instanzen von Studentengruppen überschreiben

Tabelle 3.6: Anwendungsfall “Projekt verwalten”

ID und Name	**UC1.7: Projektgruppe bilden**
Beschreibung	Benutzer bildet zusammen mit anderen Benutzern des CI-Systems eine Gruppe
Beteiligte Akteure	Akteur: Student, Student-IT und Akteur: CI-Server
Auslöser	Benutzer werden in Zukunft gemeinsam an einer Aufgabe oder einem Projekt zusammenarbeiten
Vorbedingungen	Alle Benutzer sind im System angemeldet und haben die nötigen Rechte
Nachbedingungen	Die Benutzer bilden eine Gruppe und dies wird für den Lehrbeauftragten ersichtlich
Ausnahmen	

Tabelle 3.7: Anwendungsfall “Projektgruppe bilden”

ID und Name	**UC1.8: An einem Projekt teilnehmen**
Beschreibung	Ein Student möchte an einem Projekt teilnehmen. Zuordnung wird hergestellt und der Teilnehmer hat Einsicht in die Projektdateien
Beteiligte Akteure	Akteur: Student, Student-IT und Akteur: CI-Server
Auslöser	Teilnahme an einem Projekt
Vorbedingungen	Das Projekt wurde von einem Lehrbeauftragten angelegt und freigegeben. Der Student ist Teil einer Gruppe um gemeinsamen Zugriff auf eine Projekt-Instanz zu ermöglichen
Nachbedingungen	Der Student nimmt, für den Lehrbeauftragten sichtbar, an dem Projekt teil und ist teile einer Gruppe. Es wird eine Projekt-Instanz erstellt
Ausnahmen	Studierender ist bereits diesem Projekt zugeordnet. Eine Teilnahme kann pro Person nur einmal erfolgen

Tabelle 3.8: Anwendungsfall “An einem Projekt teilnehmen”

ID und Name	**UC1.9: Einstellungen an der Projekt-Instanz vornehmen**
Beschreibung	Es werden verschiedene projektrelevante Einstellungem im CI-System vorgenommen und gespeichert
Beteiligte Akteure	Akteur: Student-IT und Akteur: CI-Server
Auslöser	Nach der initialen Einrichtung oder durch geänderte Bedingungen müssen Einstellungen angepasst werden
Vorbedingungen	Projekt-Instanz ist angelegt und die Gruppe ihrer Projekt-Instanz zugeordnet
Nachbedingungen	Einstellungen für diese konkrete Projekt-Instanz sind geändert
Ausnahmen	Einstellungen lassen sich von einem Studenten nur für die ihnen zugeordnete Projekt-Instanz ändern

Tabelle 3.9: Anwendungsfall “Einstellungen an der Projekt-Instanz vornehmen”

ID und Name	**UC1.10: Sourcen einchecken**
Beschreibung	Die IT-Studenten checken (neuen) Quellcode in die Versionierung der Projekt-Instanz ein
Beteiligte Akteure	Akteur: Student-IT und Akteur: Versionierungs-Server
Auslöser	Neuer oder veränderter Quellcode liegt vor
Vorbedingungen	Studenten sind teil einer Projekt-Instanz und einer Gruppe, der Zugang zur Versionerung wird automatisch mit dem Zugang zum CI-Server gewährt
Nachbedingungen	Der neue oder veränderte Quelltext ist im Versionierungssystem vorhanden und verarbeitet
Ausnahmen	

Tabelle 3.10: Anwendungsfall “Sourcen einchecken”

ID und Name	**UC1.11: Sourcen der Testfälle einchecken**
Beschreibung	Die IT-Studenten checken Testfälle in die Versionierung der Projekt-Instanz ein
Beteiligte Akteure	Akteur: Student-IT und Akteur: Versionierungs-Server
Auslöser	Neuer oder veränderter Quellcode liegt vor
Vorbedingungen	Studenten sind teil einer Projekt-Instanz und einer Gruppe, der Zugang zur Versionerung wird automatisch mit dem Zugang zum CI-Server gewährt
Nachbedingungen	Der neue oder veränderte Quelltext ist im Versionierungssystem vorhanden
Ausnahmen	

Tabelle 3.11: Anwendungsfall “Sourcen der Testfälle einchecken”

ID und Name	**UC1.12: Automatisches Eintragen in den Scheduler**
Beschreibung	Neue Sourcen im Versionerungssystem lösen das automatische Eintragen in den Scheduler aus, dieser überwacht das geordnete Ausführen von Build- und Testvorgang
Beteiligte Akteure	Akteur: CI-Server
Auslöser	Neuer oder veränderter wurde in das Versionierungssystem eingecheckt
Vorbedingungen	Projekt-Instanz ist angelegt
Nachbedingungen	Vorgang für diese Projekt-Instanz ist eingetragen und wartet auf Abarbeitung, siehe UC1.14
Ausnahmen	

Tabelle 3.12: Anwendungsfall “Automatisches Eintragen in den Scheduler”

ID und Name	UC1.13: Manuelles Eintragen in den Scheduler
Beschreibung	Die IT-Studenten wollen das eine Projekt-Instanz erneut gebaut und getestet wird, die Projekt-Instanz wird dazu manuell in den Scheduler eingetragen
Beteiligte Akteure	Akteur: Student-IT und Akteur: Versionierungs-Server
Auslöser	Die CI-Kette soll erneut durchgeführt werden
Vorbedingungen	Projekt-Instanz ist angelegt
Nachbedingungen	Vorgang für diese Projekt-Instanz ist eingetragen und wartet auf Abarbeitung, siehe UC1.14
Ausnahmen	

Tabelle 3.13: Anwendungsfall “Manuelles Eintragen in den Scheduler”

ID und Name	UC1.14: Buildvorgang auf dem CI-Server
Beschreibung	Die Sourcen aus dem Versionierungssystem werden im CI-System verarbeitet
Beteiligte Akteure	Akteur: CI-Server und Akteur: Versionierungs-Server
Auslöser	Neue oder veränderte Sourcen werden eingecheckt, siehe UC1.10. Der Scheduler des CI-Systems sieht den Vorgang vor. Die Eintragung kann automatisch (UC1.12) oder manuell (UC1.13) erfolgt sein
Vorbedingungen	CI-Server arbeitet und ist mit dem Versionierungsserver verknüpft
Nachbedingungen	CI-Server hat den Buildvorgang automatisch durchgeführt. Bei Erfolg wird der Projektstatus auf “Grün” gesetzt. Beim Fehlschlag wird der Status auf “Rot” gesetzt
Ausnahmen	

Tabelle 3.14: Anwendungsfall “ Buildvorgang auf dem CI-Server”

ID und Name	**UC1.15: Testvorgang auf dem CI-Server**
Beschreibung	Die zum Projekt gehörigen Testfälle werden durchgeführt
Beteiligte Akteure	Akteur: CI-Server und Akteur: Versionierungs-Server
Auslöser	Neue oder veränderte Sourcen werden eingecheckt
Vorbedingungen	UC1.14 wurde erfolgreich durchgeführt
Nachbedingungen	Bei Erfolg wird der Projektstatus auf "Grün" gesetzt. Beim Fehlschlag eines Test wird der Status auf "Rot" gesetzt
Ausnahmen	Bei Fehlschlag von UC1.14 werden keine Tests durchgeführt

Tabelle 3.15: Anwendungsfall "Testvorgang auf dem CI-Server"

ID und Name	**UC1.16: Artefakte bereitstellen**
Beschreibung	Die aus dem Buildvorgang entstandenen Artefakte werden automatisch auf dem Deployment-Server bereitgestellt
Beteiligte Akteure	Akteur: CI-Server und Akteur: Deployment-Server
Auslöser	Build- und Testvorgang waren erfolgreich und die Artefakte wurden erstellt
Vorbedingungen	Das Projekt im aktuellen Zustand produziert verwertbare Artefakte, UC1.14 und UC1.15 waren erfolgreich
Nachbedingungen	Die Artefakte sind auf dem Deployment-Server vorhanden und können abgerufen werden. Beispielhafte Artefakte sind eine installierbare Datei, ein Archiv oder eine ausführbare Datei.
Ausnahmen	Falls das Projekt noch keine verwertbaren Artefakte produziert (zum Beispiel nur Tests laufen) werden keine bereitgestellt. Bei Problemen an dieser Stelle wird der Status für das gesamte Projekt ebenfalls auf "Rot" gesetzt. Probleme die dazu führen können, dass keine Artefakte bereitgestellt werden sind unter Anderem: Die Speicherkapazität auf dem Deployment-Server ist erschöpft oder ein Ausfall der Netzwerkverbindung.

Tabelle 3.16: Anwendungsfall "Artefakte bereitstellen"

ID und Name	UC1.17: Einfache Softwaremetriken abrufen
Beschreibung	Es sollen Informationen über den Build- und Testverlauf der Projekt-Instanz angezeigt werden
Beteiligte Akteure	Akteur: Student, Student-IT, Lehrbeauftragter und Akteur: CI-Server
Auslöser	Verlauf der Projekt-Instanz ersichtlich machen
Vorbedingungen	Projekt-Instanz und Projektgruppe ist angelegt
Nachbedingungen	Metriken werden in der Webanwendung angezeigt
Ausnahmen	Wurden keine Projektdateien hochgeladen, gibt es noch keine verwertbaren Softwaremetriken. Unter den einfachen Metriken befinden sich Punkte wie der Verlauf erfolgreicher Test und erfolgreicher Build-Vorgänge

Tabelle 3.17: Anwendungsfall “Einfache Softwaremetriken abrufen”

ID und Name	UC1.18: Komplexe Testmetriken abrufen
Beschreibung	Es sollen Informationen über den Build- und Testverlauf der Projekt-Instanz angezeigt werden. Die komplexen Metriken erweitern die allen zugänglichen einfachen Metriken um konkrete Testergebnisse und weitere Codemetriken
Beteiligte Akteure	Akteur: Student-IT, Lehrbeauftragter und Akteur: CI-Server
Auslöser	Verlauf der Projekt-Instanz anhand komplexer Metriken ersichtlich machen
Vorbedingungen	Projekt-Instanz und Projektgruppe ist angelegt
Nachbedingungen	Metriken werden in der Webanwendung angezeigt
Ausnahmen	Wurden keine Projektdateien hochgeladen, gibt es noch keine verwertbaren Softwaremetriken

Tabelle 3.18: Anwendungsfall “Komplexe Testmetriken abrufen”

ID und Name	UC1.19: Abhängigkeiten bereitstellen
Beschreibung	Die für den erfolgreichen Einsatz nötigen Abhängigkeiten werden bereitgestellt, indem bei den Projekteinstellungen die Dateien ausgewählt werden
Beteiligte Akteure	Akteur: Student-IT, Akteur: CI-Server und Akteur: Deployment-Server
Auslöser	Für die erfolgreiche Installation oder den Einsatz sind weitere Bibliotheken oder andere Abhängigkeiten nötig
Vorbedingungen	Projekt-Instanz und Projektgruppe sind erstellt
Nachbedingungen	Abhängigkeiten sind auf dem Deployment-Server verfügbar
Ausnahmen	

Tabelle 3.19: Anwendungsfall “Abhängigkeiten bereitstellen”

ID und Name	UC1.20 Artefakte und Abhängigkeiten abrufen
Beschreibung	Die sich aus einer Projekt-Instanz ergebenen Artefakte werden bereitgestellt und ein Studierender möchte diese herunterladen und benutzen
Beteiligte Akteure	Akteur: Student, Student-IT, Lehrbeauftragter und Akteur: Deployment-Server
Auslöser	Herunterladen der fertigen Artefakte der Projekt-Instanz
Vorbedingungen	Projektdateien wurden hochgeladen und erfolgreich gebaut, Artefakte sind vom CI-Server auf den Deployment-Server geschoben worden
Nachbedingungen	Der Studierende hat lokal Zugriff auf alle nötigen Dateien um die Software zu verwenden
Ausnahmen	Student muss Teil der Gruppe beziehungsweise der Veranstaltung sein, um Zugriff auf Artefakte zu erhalten. Im Fehlerfall werden diese nicht sichtbar

Tabelle 3.20: Anwendungsfall “Artefakte und Abhängigkeiten abrufen”

3.3 Nichtfunktionale Anforderungen

Abseits der zur Verfügung stehenden Funktionen, muss ein solches Softwaresystem noch weitere, wichtige Kriterien erfüllen, um als Ganzes akzeptiert zu werden. Nichtfunktionale Anforderungen legen dabei Wert darauf, wie etwas umgesetzt wird, und weniger auf den technischen Hintergrund. Im Folgenden werden eine Reihe charakteristischer nichtfunktionaler Anforderungen spezifiziert und auf die Anwendung bezogen beschrieben.

Zuverlässigkeit: Fehlerhafte Eingaben sollen mit allen Mitteln verhindert, beziehungsweise minimiert werden. Das Interface stellt nur die Funktionen zur Verfügung, die von einer Benutzergruppe unbedingt benötigt werden. Zusätzlich werden viele Abläufe automatisiert um Fehler durch Falscheingaben auszuschließen. Im Betrieb wird von einem sehr geringen Wartungsaufwand beim Server ausgegangen, dazu muss man sicherstellen, dass auftretende Fehler nicht zu einem Absturz oder Systemstillstand führen und selbstständig korrigiert werden können. Informations- oder Warnmeldungen geben dem Benutzer Auskunft über Eingabeirrtümer und Hinweise zur Korrektur selbiger.

Aussehen und Handhabung: Besonderer Wert muss auf eine möglichst einfache und benutzerfreundliche Handhabung gelegt werden. Da auch Personenkreise mit geringerem technischen Verständnis das System nutzen werden, ist auf eine intuitive oder leicht erlernbare Benutzerführung zu achten. Ziel ist aktuelle GUI-Richtlinien umzusetzen und damit den Einstieg oder Umstieg zu erleichtern. Starke Kontraste und die Nutzung von Farbschemas, welche auch auf anderen Webseiten oder Portalen der HAW Hamburg Anwendung finden, erhöhen den Wiedererkennungs- und Identifikationswert der Anwendung.

Leistung und Effizienz: Da die technische Infrastruktur für ein System in diesem Umfang mehr als ausreichend ist, kann man diesen Punkt im Vergleich zu anderen vernachlässigen. Besonders die projekt- und veranstaltungsorientierte Einrichtung mehrerer voneinander getrennter Server führt dazu, dass man pro System von einer gut abschätzbaren Anwenderanzahl ausgehen kann. Festplattenkapazität ist für die zu erwartenden Studentenzahlen in unterschiedlichen Projekten und Veranstaltungen gebührend vorhanden und kann genutzt werden. Man skaliert die gesamte Anwendung für einen 1.5x-2x größeren Andrang als initial abgeschätzt um auf alle Eventualitäten vorbereitet zu sein.

Wartung, Änderbarkeit und Erweiterbarkeit: Beim Systemdesign muss auf starke Modularität der Komponenten wert gelegt werden, da sich stetig ändernde Anforderungen neue Schwerpunkte mit sich bringen, auf die reagiert werden muss. Für den Einsatz in zukünftigen Semestern muss eine gute Dokumentation und Beschreibung der Module sicherstellen, dass man das bestehende System sicher erweitern und ändern kann. Eine klare Abgrenzung der einzelnen Komponenten führt zu Entkoppelung der Serverkomponenten und fördert die Umstellung auf einen anderen Fokus.

Portierbarkeit: Es muss ein möglichst weit verbreitetes System gewählt werden, welches eine Vielzahl von Erweiterungen unterstützt, damit das System auch in sich verändernden Kontexten weiterhin problemlos eingesetzt werden kann.

Datenschutz und Sicherkeit: Die sensiblen Daten der Studierenden und Lehrbeauftragen müssen sicher abgelegt und verwendet werden. Die Projektdaten und Bibliotheken sind vorerst nur im internen Netz der HAW Hamburg verfügbar und damit ausreichend von einem unautorisierten Zugriff von außen geschützt. Intern wird eine Authentifizierung am System genutzt und Zugriff auf Projektdaten zu gewähren. Die fertiggestellten Artefakte liegen auf dem lokalen System oder auf einem weiteren entfernt stehenden Server.

3.4 Zusammenfassung der Ergebnisse

Eine CI-Plattform an einer Hochschule, besonders wenn Studierende unterschiedlicher Fachrichten zusammenkommen, bedeutet auch einen erhöhten Aufwand bei der Analyse und dem Systementwurf. Unterschiedliche Fachrichten und Veranstaltungen wünschen die Unterstützung vielfältiger Programmiersprachen. Eine Plattform zu finden, die alle Wünsche erfüllt bedeutet schon eine erste Hürde. Ebenso bedeuten die unterschiedlichen Anwendungsschwerpunkte bei den Studierenden, dass die Plattform leicht zu erlernen und zu benutzen sein muss. Bedingt durch die sich stetig ändernden Bedingungen an einer Hochschule, insbesondere bei der Informatik, muss das System stark auf Anpassbarkeit ausgelegt sein. Sich ändernde Veranstaltungsschwerpunkte, neue Projekte und neue Lehrbeauftragte erfordern eine Umsetzung durch die CI-Plattform.

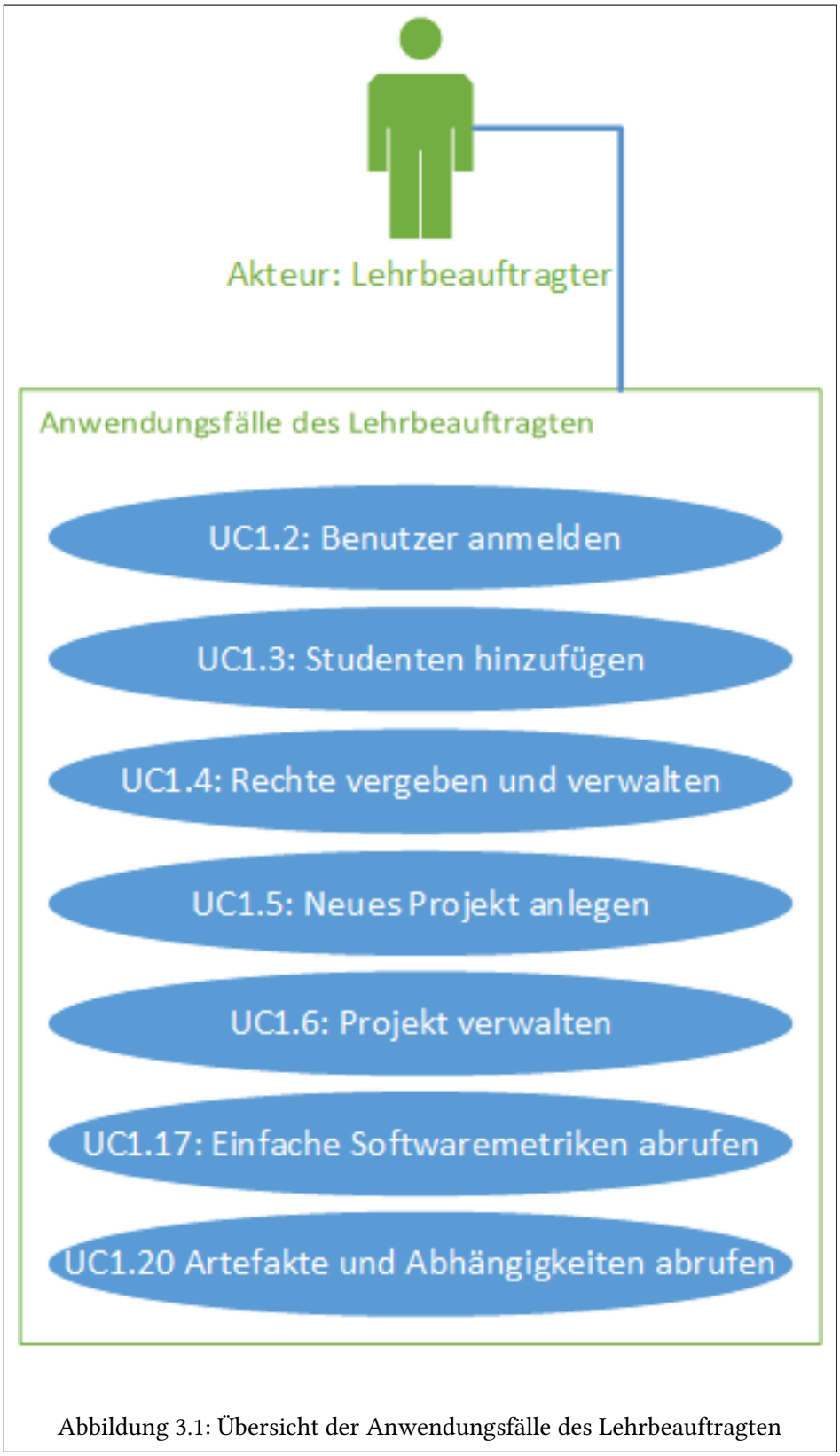


Abbildung 3.1: Übersicht der Anwendungsfälle des Lehrbeauftragten

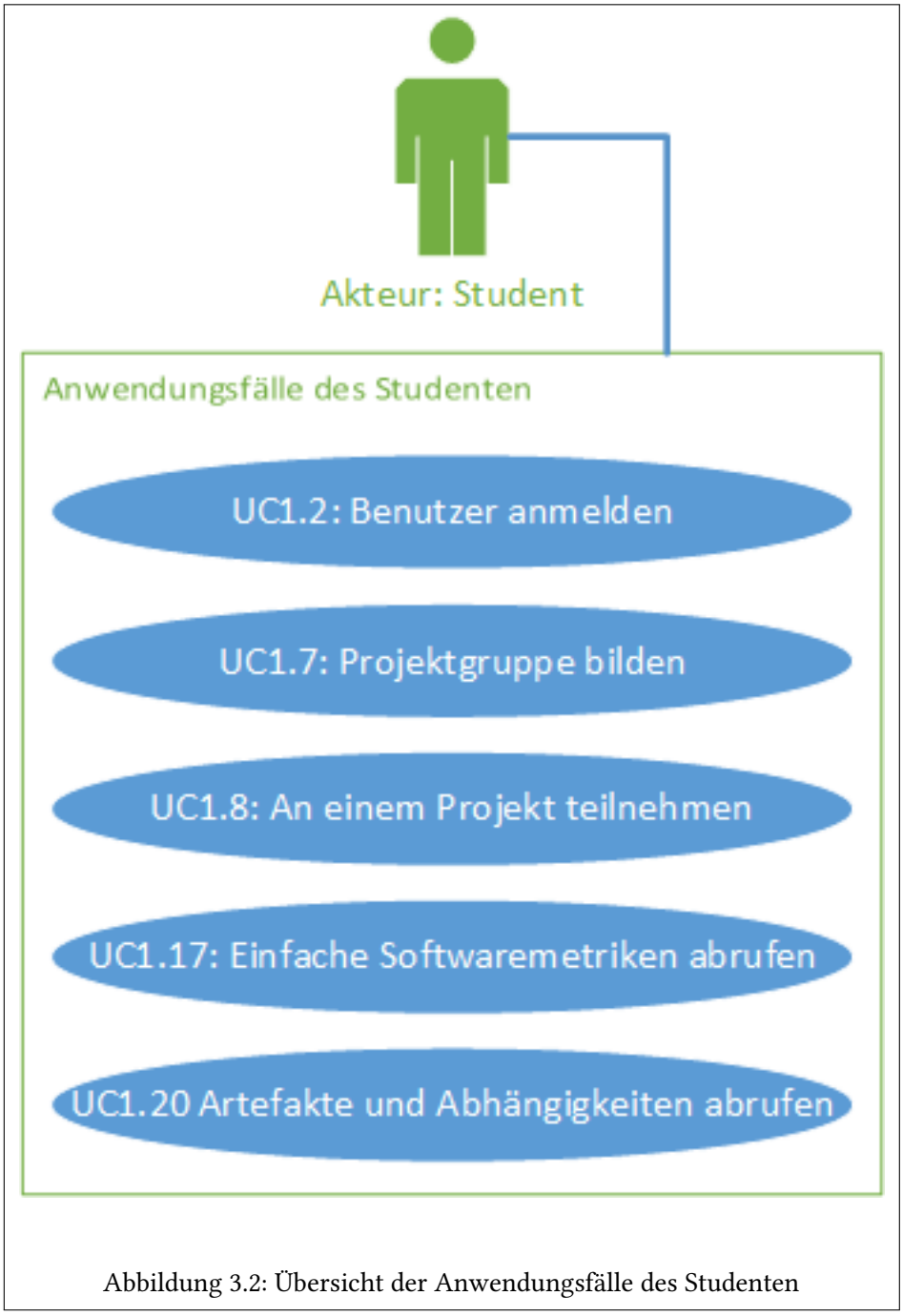


Abbildung 3.2: Übersicht der Anwendungsfälle des Studenten

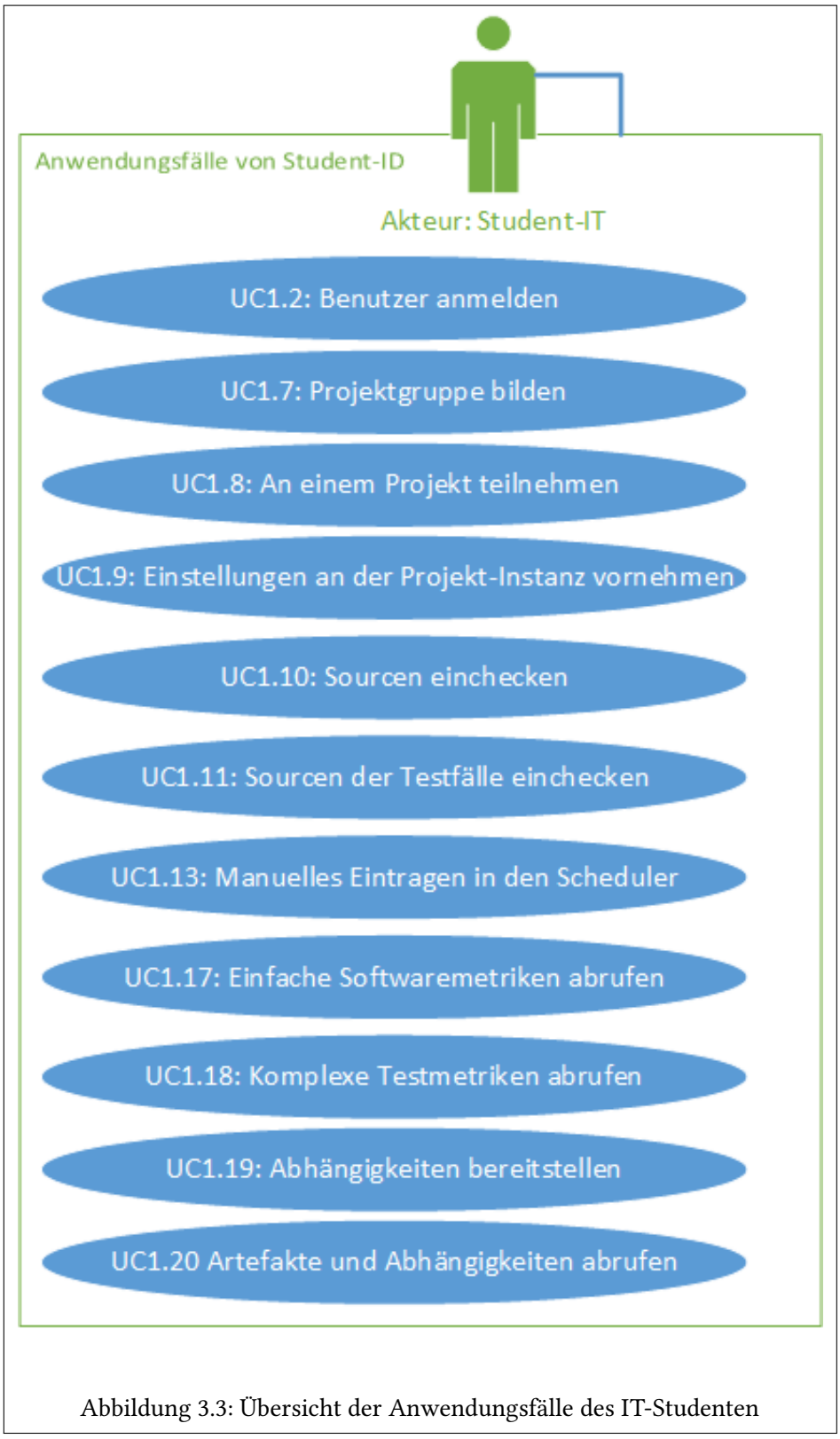


Abbildung 3.3: Übersicht der Anwendungsfälle des IT-Studenten

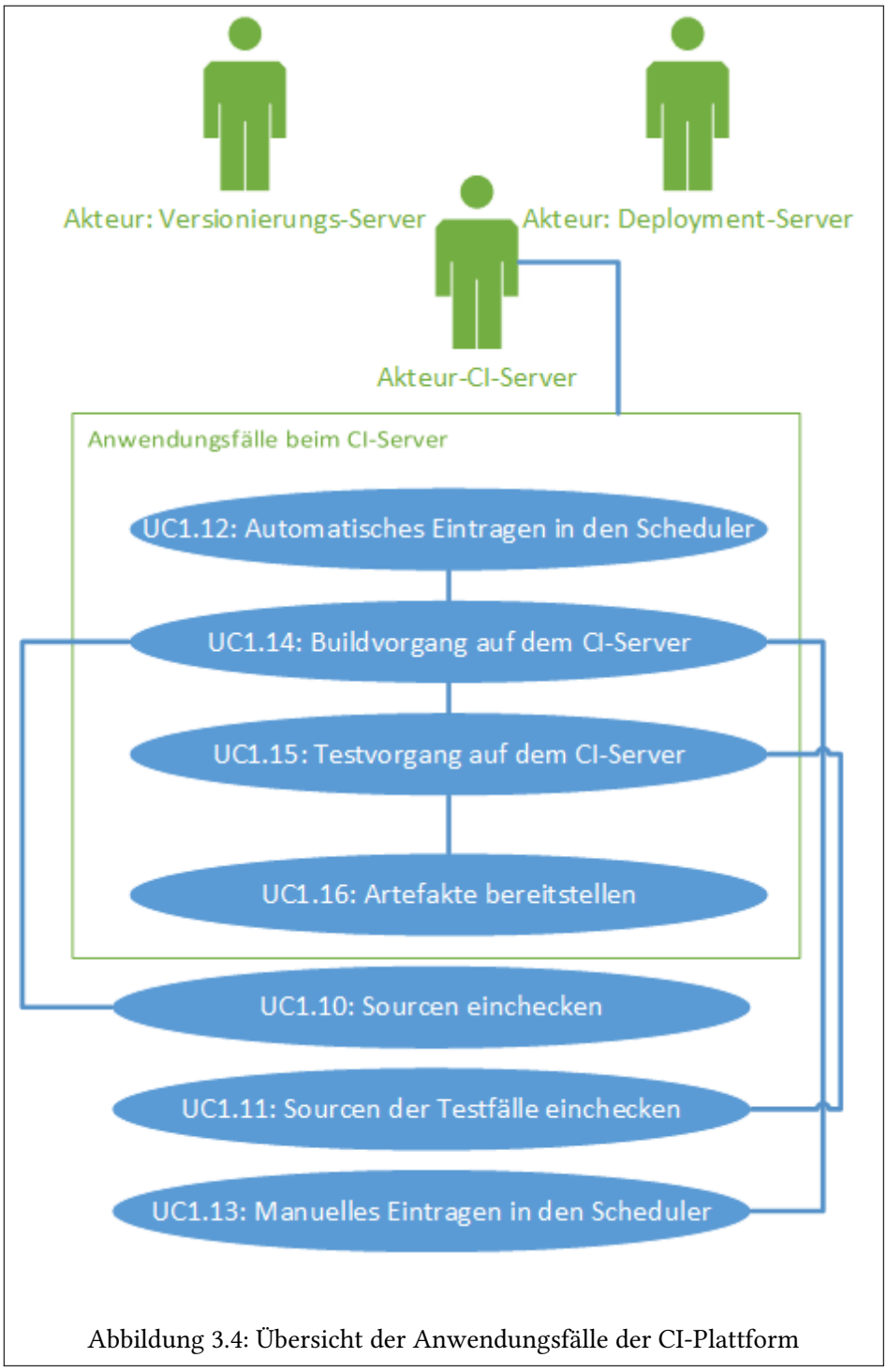


Abbildung 3.4: Übersicht der Anwendungsfälle der CI-Plattform

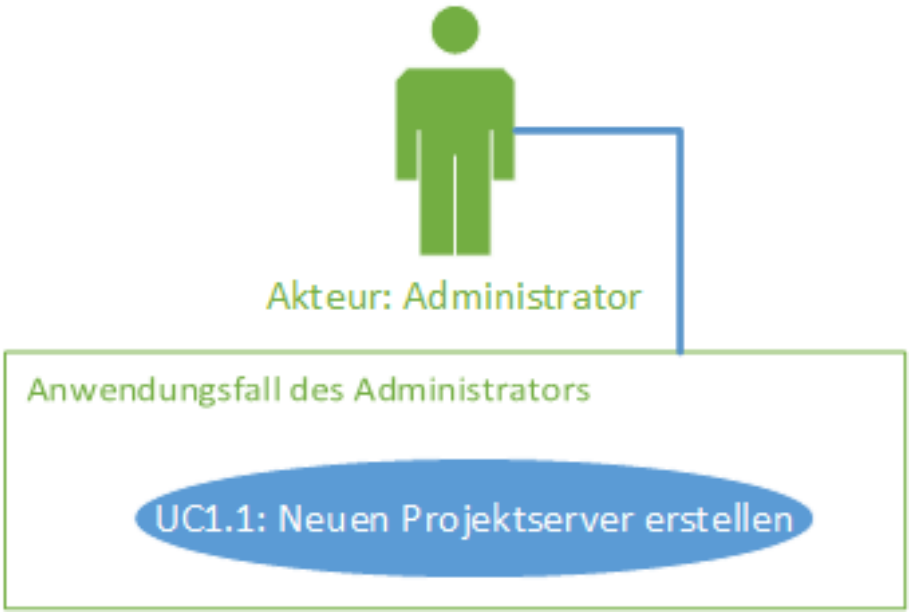


Abbildung 3.5: Übersicht der Anwendungsfälle des Administrators

4 Continuous Integration Systeme

In diesem Kapitel werden verschiedene Continuous Integration Systeme untersucht. Anhand der im vorherigen Kapitel erarbeiteten Anforderungen an ein solches System, werden diese für den Einsatz an einer Hochschule bewertet. Aufgrund der hohen Anzahl an verfügbaren Systemen und des daraus resultierenden Umfangs, wird sich auf eine Auswahl von Systemen konzentriert und einige weitere am Ende kurz abgehandelt. Wichtig für einen frei konfigurierbaren Ansatz ist in erster Linie ein Open-Source System. Diese bieten, durch viele Erweiterungen der Community und die Mittel, selber zusätzliche Features zu entwickeln, eine breiteres Spektrum an Möglichkeiten, als eine proprietäre Lösung.

4.1 Software

Vor einigen Jahren gab es nur eine Open-Source Lösung, die bereits umfangreich genug war, den kommerziellen Lösungen in nicht allzu vielen Punkten nachzustehen. CruiseControl, ursprünglich von Angestellen der Firma ThoughtWorks entwickelt und später als eigene Anwendung ausgegliedert. In dieser Firma arbeitet auch einer der Entwickler von Continuous Integration, Martin Fowler.

Seitdem haben sich aber eine Reihe weiterer Applikation auf dem Markt etabliert und sind im Umfang gewachsen. Viele Lösungen stehen den Bezahlprodukten in nur noch wenigen oder keinen Punkten mehr nach und bieten den persönlichen Bedürfnissen entsprechende Lösungen.

Jenkins, ein Ableger von Hudson, ist eine der am weitesten verbreiteten Open-Source CI-Anwendung ist. Eine große Gemeinschaft an Entwicklern die Plugins schreiben und pflegen bestätigt diese Aussage. Da der Einsatz am sichersten mit einer sich stetig weiterentwickelnden Lösung ist, wird in diesem Kapitel Jenkins tiefergehend betrachtet, als direkter Konkurrent und erstes Open-Source Tool zusätzlich CruiseControl. Um die proprietären Lösungen nicht außer Acht zu lassen, wird am Beispiel vom Microsoft Team Foundation Server eine vergleichende Betrachung zu den Open-Source Systemen getätigt.

Weitere Lösungen, die aber nicht den Verbreitungsgrad von Jenkins erreichen, werden im Anschluss ebenfalls erläutert.

4.1.1 Entscheidungskriterien für ein System

Aufgrund der großen Fülle an bereits etablierten proprietären und Open-Source CI-Systemen auf dem Markt lässt sich die erste Frage: “Macht die Neuentwicklung eines CI-Servers für Hochschulen Sinn?” mit einem klaren Nein beantworten. Sämtliche Anforderungen werden bereits durch Lösungen abgedeckt beziehungsweise lassen sich durch Modifikation und Erweiterung bestehender Plattformen realisieren.[1]

Es ist also nötig, sich klar zu machen, welche Kriterien man der Auswahlentscheidung zu Grunde legen muss. Diese werden für den konkreten Anwendungskontext gewählt, da die Lösungen zwar viele Bereiche abdecken, aber es keine Plattform gibt, die alles realisiert.

- Kommt der Kauf einer proprietären Lösung in Frage oder sollte man auf Open-Source setzen? Der allgemeine Konsens unter Entwicklern ist, dass je nach Anwendungsbereich, die Open-Source Lösungen ihren kaufbaren Gegenstücken in nichts oder nur wenigen Punkten nachstehen. Die Frage für Hochschulen sind dann Lizenzkosten (gibt es eine Möglichkeit Lizenzen für den Lehrbetrieb zu erhalten?) und ob ein fester Support gewünscht wird. Andererseits bieten quelloffene Lösungen eine aktive Community die Erweiterungen bereitstellt und Ansatzpunkte sogar eigene Modifikationen vorzunehmen.
- Welche Funktionen gefordert werden spielt eine wichtige Rolle. Die Grundfunktionalität bei Open-Source Lösungen lässt sich oftmals durch Plugins erweitern. Besonders im Kontext der Hochschule kommen oftmals eine Vielzahl unterschiedlicher Programmiersprachen und Zielsysteme zum Einsatz, die eine Fixierung auf nur eine oder wenige Sprachen von einem System von vorneherein ausschließen. Damit ist der Punkt der Erweiterbarkeit beziehungsweise des grundlegenden Funktionsumfangs einer der wichtigsten.
- Ob es eine vorhandene Infrastruktur gibt, spielt bei vielen Systemen eine wichtige Rolle. Da in dieser Arbeit davon ausgegangen wird, dass der aktuelle Stand eine Neueinführung einer solchen Plattform ist, muss die bisherige Infrastruktur nicht berücksichtigt wer-

[1] http://en.wikipedia.org/wiki/Comparison_of_continuous_integration_software

den. Studierende und Lehrbeauftragte müssen den Umgang mit einem solchen System erlernen, da es zum momentanen Zeitpunkt keine alternative Lösung gibt.

- Der letzte Punkt führt auch zu einem weiteren wichtigen Kriterium: “Wie leicht ist der Umgang mit der CI-Plattform zu erlernen?”. Viele Schritte sollten durch eingängige Menüs abgenommen werden, so dass der Verwaltungsaufwand und manuelle Aufwand möglichst gering bleibt. Eine ausführliche Dokumentation der eingesetzten Software erleichtert den Einstieg.

Zu einem ähnlichen Schluss der Kriterien kommt auch der Artikel auf der Technik-Seite heise.de [Bjoern Feustel (2012)] und Paul M. Duvall in seinem Buch[Paul M. Duvall (2007)].

4.1.2 Entscheidungskriterien für Open-Source

Besonders im Umfeld einer Hochschule macht der Einsatz von Open-Source Software Sinn, da damit einher viele Vorteile und Möglichkeiten gehen, die eine eingekaufte Lösung nicht bietet. Mit den Entscheidungsfaktoren zum Einsatz von Open-Source hat sich bereits Prof. Dr. Michael Gröschel der Hochschule Mannheim befasst und in einem Artikel zusammengefasst. [Groeschel (2012)]

- Vielfältige Quellen unterstützen ein Open-Source System. Angefangen bei den Entwicklern, über die Community, bis hin zu Dienstleistern, die kundengerechte Anpassungen gegen Entgelt bieten.
- Der offengelegte Quelltext bietet die Möglichkeit kleine oder größere Änderungen durchzuführen. An einer Hochschule natürlich nur im Rahmen der gegebenen Möglichkeiten.
- Die Unabhängigkeit von einem Softwarehersteller senkt Kosten oder eliminiert diese komplett. Bei verbreiteten Systemen ist lange in die Zukunft ein Support gewährleistet.
- Eine große Community steigert oftmals auch die Softwarequalität und Fehler können durch kürzere Patchzyklen schneller beseitigt werden.
- Der wirtschaftliche Faktor durch Open-Source ist nicht zu vernachlässigen. Keine zu erneuernden Lizenzen oder Gebühren für Erweiterungen fallen an.
- Bringt sich eine Hochschule aktiv in eine Open-Source Anwendung ein, führt dies häufig zu einer sich selbst verstärkenden Wechselwirkung, die für alle Seiten positiv

ausfällt. Wachsende Communities bedeuten immer einen Mehrwert bei der Open-Source Entwicklung.

- Offene Schnittstellen, wie sie auch Anwendung in Open-Source Lösungen finden, werden häufig weiter genutzt um neue Plattformen zu erschließen (Web, Mobile Geräte)

4.1.3 Jenkins

Der CI-Server Jenkins [Web03] entstand am 29.01.2011 aus dem Hudson-Projekt, nachdem eine Übernahme von Oracle die Community dazu bewogen hat eine Namensänderung zu vollziehen. Im Rahmen der Übernahme hat die Firma Oracle den Namen Hudson rechtlich schützen lassen, was dazu führte das in der hinter dem Projekt stehenden Gemeinschaft zuallererst eine Änderung zu Jenkins vorgeschlagen wurde. Das Endresultat war dann, dass nicht nur Jenkins entwickelt wurde, sondern auch parallel Hudson und somit Jenkins als Fork anzusehen ist.

Bereits über Hudson wurde geschrieben, warum noch kein Umstieg von der ersten richtigen CI-Software "CruiseControl" stattgefunden hat. [Dyer (2008)]. Der Bekanntheitsgrad von Jenkins nimmt weiter zu und am 10.12.2013 hat Jenkins auf Github 567 Mitglieder und 1100 öffentliche Repositorien[2]. Im Gegensatz dazu besitzt Hudson nur 32 Mitglieder und 17 öffentliche Repositorien[3]. Im Jenkins-Wiki steht eine respektable Liste von Unternehmen und Open-Source Projekten, welche die Software einsetzen[4].

Hudson, wie dann auch Jenkins, sind in Java geschrieben, der ursprüngliche Entwickler ist Kohsuke Kawaguchi, welcher auch weiter daran arbeitet. Es ist ein serverbasiertes, webbasiertes Tool, welches in der Grundversion viele Versionierungen unterstützt und Apache Ant sowie Apache Maven Projekte bauen kann. In erster Linie dient Jenkins dazu Java-Projekte zu verwalten, durch die sehr gute Erweiterbarkeit können aber eine beachtliche Anzahl weiterer Programmiersprachen und damit Projekte verwaltet werden. Zentrales Element ist ein übersichtliches Dashboard, von dem die Benutzer die Projekte steuern können. Durch Ansichten lässt dieses sich weiter personalisieren.

Weitere Elemente von Jenkins sind die Möglichkeit verteilte Builds[5] durchzuführen, so lässt sich die Last sehr gut verteilen und das System den eigenen Bedürfnissen anpassen. Bereits

[2] https://github.com/jenkinsci (10.12.2013)
[3] https://github.com/hudson (10.12.2013)
[4] https://wiki.jenkins-ci.org/pages/viewpage.action?pageId=58001258 (10.12.2013)
[5] https://wiki.jenkins-ci.org/display/JENKINS/Distributed+builds (10.12.2013)

ohne eine Master/Slave Konfiguration bieten die Einstellungen bei den Jobs die Möglichkeit auf die Gesamtlast des Systems Rücksicht zu nehmen. Fortgeschrittenen Einstellungen und Benutzungsszenarien von Jenkins wurde ein eigenes Buch gewidmet. [Berg (2012)]
Sowohl Software selber, als auch der Quelltext sind frei einsehbar und wiederverwendbar. Die verwendete Lizenz ist die MIT-Lizenz[6].

4.1.4 CruiseControl

CruiseControl [Web02] wird an dieser Stelle erwähnt, da es sich dabei um die erste richtige Software für CI handelt. Die ursprüngliche Version und damit der erste Quellcode wurde von der Firma ThoughtWorks[7] für den Gebrauch innerhalb des Unternehmens geschrieben. Zu einem späteren Zeitpunkt wurde CruiseControl in eine eigene Anwendung ausgegliedert.

Wie auch Jenkins unterstützt CruiseControl eine große Zahl an Versionierungs-Tools, lässt sich aber schlechter auf weitere Programmiersprachen erweitern. Das Programm selber ist in Java geschrieben und dient so auch der Verwaltung von Java-Projekten. Um andere Sprachen und Projekte zu verwalten, ist der Einsatz einer angepassten Version erforderlich. Für Ruby ist dies CruiseControl.rb[8] und für .NET CruiseControl.net[9].

Die Software besteht aus zwei Bestandteilen, einmal der “Build Loop” und das “Status Dashboard”. Ersteres läuft als Deamon, das Dashboard ist webbasiert. Die Funktion lässt sich durch Plugins erweitern und es gibt eine Reihe von Schnittstellen zu Nachbarsystemen, wie der Versionierung oder dem Build-Management. Es gibt zwar auch eine ganze Menge an Plugins, CruiseControl steht aber, trotz der längeren Präsenz am Markt, hinter Jenkins zurück[10]. Das Dashboard kam dabei erst zu einem späteren Zeitpunkt der Entwicklung dazu und ist nicht so weit fortgeschritten wie das von Jenkins, welches von vorneherein ein zentraler Bestandteil war.

Die verwendete Software-Lizenz ist eine an die BSD-Lizenz[11] angelehnt.

[6] http://de.wikipedia.org/wiki/MIT-Lizenz (20.12.2013)
[7] http://www.thoughtworks.com/ (10.12.2013)
[8] http://cruisecontrolrb.thoughtworks.com/ (10.12.2013)
[9] 10.12.2013)
[10] http://cruisecontrol.sourceforge.net/main/plugins.html (10.12.2013)
[11] http://en.wikipedia.org/wiki/BSD_license (10.12.2013)

4.1.5 Microsoft Team Fundation Server

Um die proprietären Lösungen nicht außer acht zu lassen, wird der Microsoft Team Foundation Server[12] [Web01] einer Analyse unterzogen. Im Kern deckt der Team Foundation Server alle Elemente von Continuous Integration ab. Es bietet Versionierung (Git oder die Team Foundation Version Control), eine Projektverwaltung, automatisierte Builds, Testing und Release Management (Deployment). TFS ist ein hochintegrierter Teil von Visual Studio und wird in diesem Zusammenhang als CI-Server eingesetzt, es kann aber auch zusammen mit Eclipse und damit Plattform-unabhängig eingesetzt werden[13].

Es gibt eine sehr umfangreiche Dokumentation zu TFS, welche alle Bereiche von der Einrichtung bis zur Benutzung abdeckt. Zum Beispiel die einfache[14] und komplexe[15] Konfiguration. Tutorials beschreiben die nötigen Einstellungen für CI mit TFS[16]. Eine Reihe von Büchern befassen sich ebenfalls mit dem Thema, beispielhaft ist hier "Visual Studio Team Foundation Server 2012: Adopting Agile Software Practices, From Backlog to Continuous Feedback" von Sam Guckenheimer und Neno Loje zu nennen [Sam Guckenheimer (2013)], welches auf TFS in Zusammenhang mit agiler Softwareentwicklung eingeht.

Die einzelnen Elemente für das Testing, Reporting sind von TFS vorgegeben und lassen sich nicht den eigenen Bedürfnissen anpassen. Es bietet aber APIs, durch die Java oder .NET Clients auf die Funktionen zugreifen können[17].

Der Team Foundation Server steht unter der Microsoft EULA.

4.1.6 Weitere Systeme

Erwähnenswert sind mehrere online bereitgestellte Continuous Integration Server wie zum Beispiel BuildHive[18], CircleCi[19] und CiSimple[20], mit respektablem Funktionsumfang, aber zu einschränkt für die angestrebte Umsetzung. Weitere Open-Source Lösungen sind Travis

[12] im Folgenden auch TFS
[13] http://msdn.microsoft.com/en-us/library/gg413285.aspx (10.12.2013)
[14] http://msdn.microsoft.com/en-us/library/vstudio/ee259686%28v=vs.120%29.aspx (10.12.2013)
[15] http://msdn.microsoft.com/en-us/library/vstudio/dd631919%28v=vs.120%29.aspx (10.12.2013)
[16] http://msdn.microsoft.com/en-us/library/hh395023%28v=vs.110%29.aspx (10.12.2013)
[17] http://msdn.microsoft.com/en-us/library/bb130146.aspx (10.12.2013)
[18] https://buildhive.cloudbees.com/ (20.11.2013)
[19] https://circleci.com/ (20.11.2013)
[20] https://www.cisimple.com/ (20.11.2013)

CI[21] und Apache Continuum[22], wobei ersteres für GitHub genutzt wird und letzteres für Java-Projekte zusammen mit Maven. TeamCity[23] ist eine weitere proprietäre Software für CI. Es gibt noch weitere Alternativen, eine Übersicht ist auf Wikipedia[24] zu finden.

4.2 Bewertung der Systeme

Durch Plugins lässt sich bei Jenkins-CI der größte Funktionsumfang herstellen und am einfachsten Modifikationen vornehmen. Eine proprietäre Lösung wie TFS oder TeamCity wurde von vorneherein ausgeschlossen und die Recherche hat ergeben, dass der Funktionsumfang von Open-Source Systemen diesen auch in nichts nachsteht. Ebenso sind online bereitsgestellte CI-Server nicht optimal für den Einsatz an einer Hochschule geeignet. Eine sehr ausführliche Übersicht über verschiedene CI-Server und deren Funktionalität ist im Internet bei ThoughtWorks zu finden [Web13 (2013)]. Leider ohne Microsoft Team Foundation Server und Jenkins (dafür aber mit Hudson).

[21] https://travis-ci.org/ (20.11.2013)
[22] http://continuum.apache.org/ (20.11.2013)
[23] http://www.jetbrains.com/teamcity/ (20.11.2013)
[24] http://en.wikipedia.org/wiki/Continuous_integration (20.11.2013)

5 Serverumgebung

Aus den Anforderungen und den daraus resultierenden Anwendungsfällen in Kapitel Drei, ergibt sich ein Mindestumfang der zu erstellenden CI-Plattform. Im Produktivbetrieb sieht diese Umgebung dann wie in Abb. 5.1 aus. Dieses Kapitel hat als Schwerpunkt die Integration der Module in Jenkins und die Darstellung der zusammenhängenden Systemarchitektur.

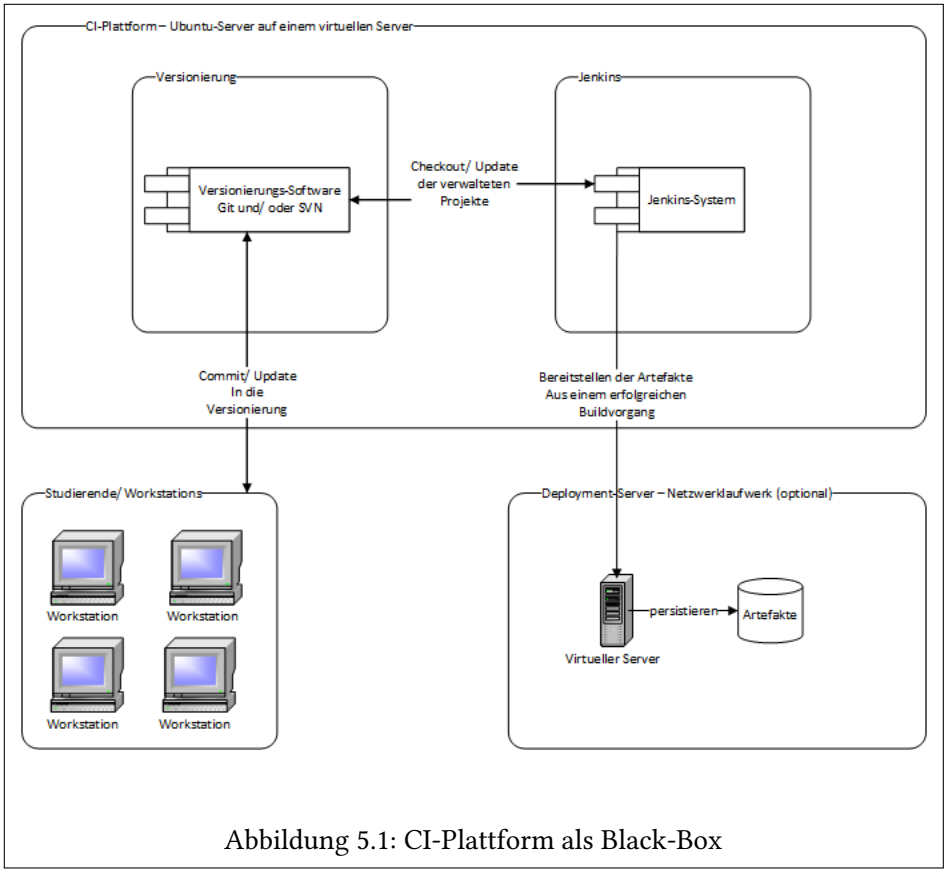


Abbildung 5.1: CI-Plattform als Black-Box

5.1 Systemarchitektur von Jenkins

5.1.1 Beschreibung der Systemarchitektur und Kontext

Bevor diese Arbeit nun auf die fertige Systemarchitektur der angepassten CI-Plattform Jenkins eingeht, folgt die Einordnung in den, in der Einleitung festgelegten, Kontext (Abb. 5.1).

Es gibt eine virtuellen Server mit dem Ubuntu-Server Betriebssystem, auf diesem läuft eine Jenkins-Instanz mit allen Modulen, der Versionierungssoftware und allen weiten benötigten Softwarepaketen für den Betrieb. Über einen Netzwerkzugriff checken die Studierenden ihren Code in die Versionierung ein, diese wird periodisch von Jenkins auf Änderungen überprüft. Innerhalb von Jenkins findet der Build- Test- und Deployment-Vorgang statt. Der letzte Schritt ist die Auslieferung der erstellten Artefakte an einen Ort für das Deployment. Ein weiterer entfernter Server ist optional, alternativ kann auch ein Ort auf der lokalen Instanz als Ziel ausgewählt werden. In der Regel ist es aber an der Hochschule von Vorteil, wenn es einen persistenten Server gibt, auf dem alle, im Rahmen einer Veranstaltung erstellten, Artefakte vorgehalten und bereitgestellt werden.

Ohne genauer auf konkrete Implementierungen einzugehen, zeigen die folgenden Diagramme (Abb. 5.2, Abb. 5.3, Abb. 5.4) alle Akteure, Abhängigkeiten und Applikationselemente auf.

Der linke Teil der Abbildung (Abb. 5.2) sind die Studenten und Lehrbeauftragten, die über ein Webinterface, beziehungsweise die Versionierung, mit der CI-Plattform interagieren. Das Frontend wird von Jenkins bereitgestellt und ermöglicht Zugriff auf die Funktionen. Zwischen Studierenden und Lehrbeauftragten besteht eine Verbindung über gemeinsame Projekte und Lehrveranstaltungen. Ferner gibt es einen Administrator, welcher die Einrichtung der CI-Plattform vornehmen muss.

Der zentrale Teil der Abbildung (Abb. 5.3) ist der CI-Server (Jenkins) und die Versionierung. Nach außen hin wird über die verschiedenen Frontends und die Verbindung zur Versionierung kommuniziert. Eine Vielzahl von Plugins in Jenkins ermöglicht die verschiedenen Funktionen. Eine weitere Verbindung, lokal oder über ein Netzwerk, besteht zu einem Speicherort für die kompilierten Artefakte.

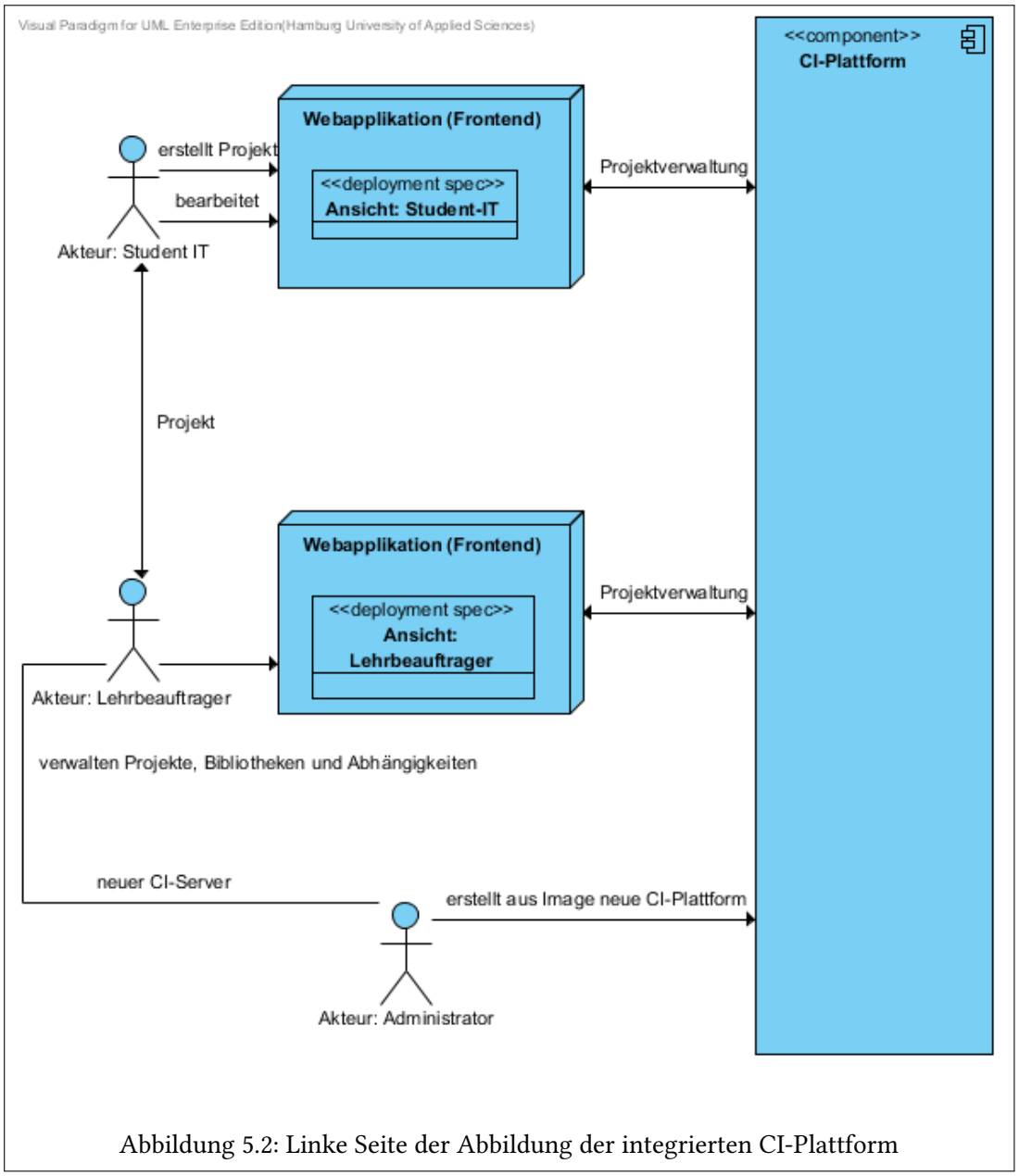


Abbildung 5.2: Linke Seite der Abbildung der integrierten CI-Plattform

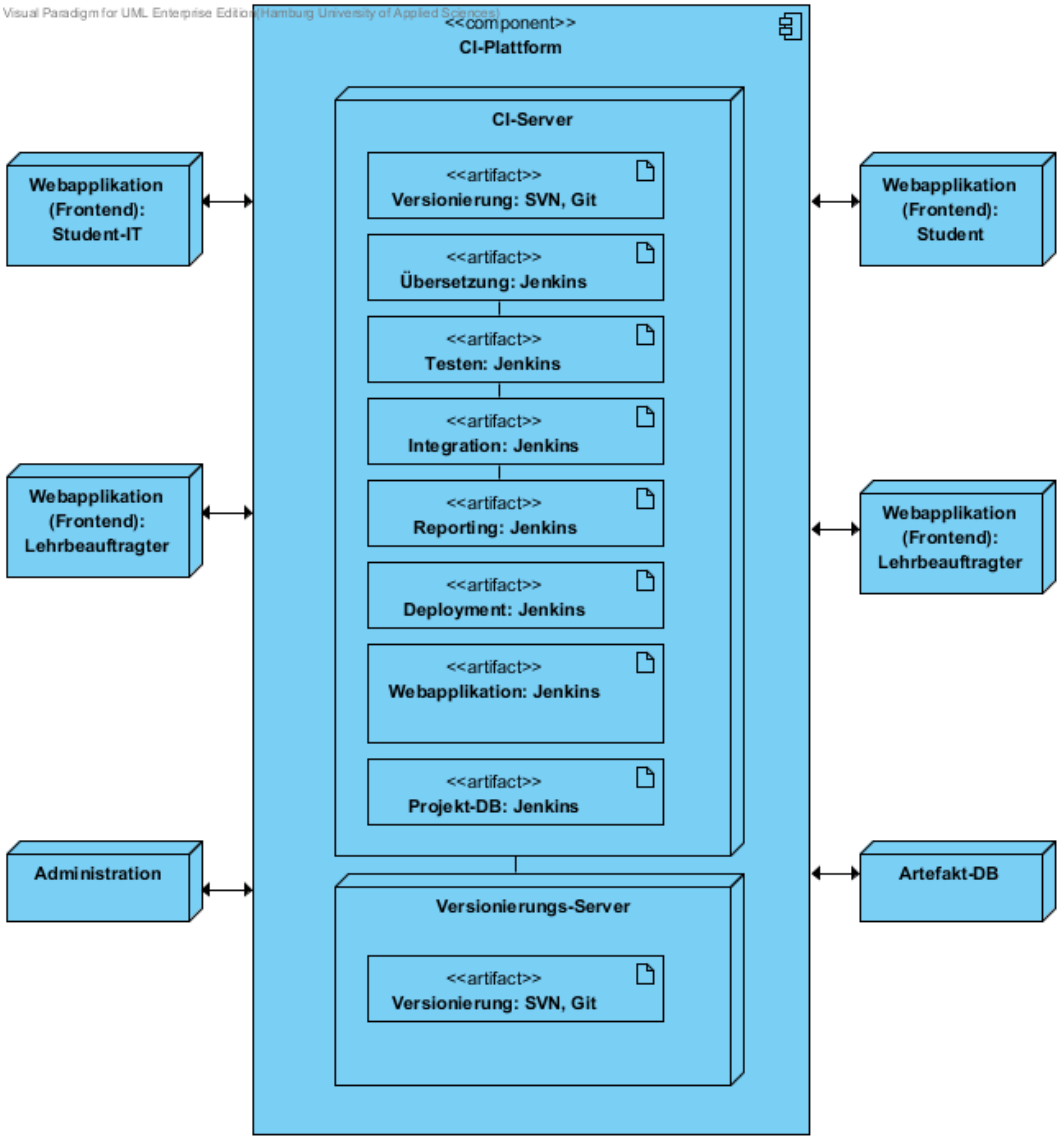


Abbildung 5.3: Mittlerer Teil Abbildung der integrierten CI-Plattform

Der rechte Teil der Abbildung (Abb. 5.4) zeigt die Frontends der Studierenden ohne IT-Hintergrund, nochmals die Lehrbeauftragten und den Ablageort der erstellten Artefakte. Die Verbindungen bestehen zur CI-Plattform.

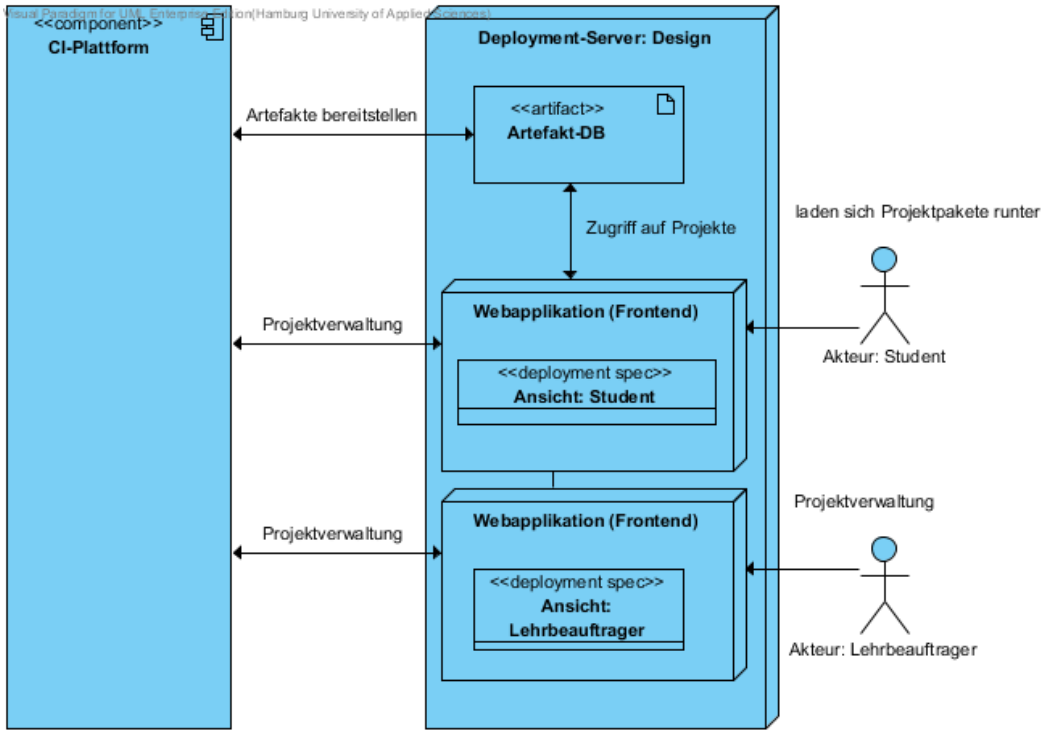


Abbildung 5.4: Rechte Seite der Abbildung der integrierten CI-Plattform

Die verschiedenen Akteure greifen über, für den speziellen Anwendungskontext definierte, Schnittstellen (Ansicht in der Webapplikation, administrative Sicht auf den virtuellen Server) auf die Anwendung zu. Die Interaktionen sind im Detail in Kapitel Drei ab Seite 14 in Form der Anwendungsfälle beschrieben, in der Grafik werden die Abhängigkeiten untereinander deutlich. Es zeichnet sich bereits ab, dass CI-Server, Webapplikation und Deployment-Server wiederrum aus mehreren Komponenten bestehen, die eng verzahnt miteinander arbeiten.

5.2 Auswahlentscheidungen

In diesem Abschnitt werden die Auswahlentscheidungen aus Kapitel 4 konkretisiert, dessen Ergebnis war, die Open-Source-Plattform Jenkins zu nutzen. Es ist nötig sich auf eine Versionierungs-Software festzulegen, mit denen der CI-Server kommuniziert. Plugins, die sicherstellen, dass alle benötigsten Programmiersprachen unterstützt werden, müssen ausgewählt, initial eingerichtet und getestet werden. Direkt in Abhängigkeit dazu, gibt es zusätzliche Erweiterungen für Code-Metriken und Reporting, die installiert werden. Die Verknüpfung mit einem entfernt oder lokal stehenden Deployment-Server, für die abzuliefernden Artefakte der einzelnen Software-Projekte, wird hergestellt. Als abschließender Teil müssen die Benutzerrollen umgesetzt und in eine Benutzerdatenbank mit Rechteverwaltung abgelegt werden. Die am System stattfindende Autorisierung und Authentifizierung findet ebenfalls über Bestandteile von Jenkins statt.

Bevor man sich den eigentlichen Module in Jenkins widmet, müssen einige Entscheidungen bezüglich der zu verwendenden Komponenten getroffen werden. Im Folgenden begründe ich die getroffenen Entscheidung für den speziellen Anwendungsfall, der in dieser Arbeit vorliegt.

5.2.1 Installation von Plugins

Nachfolgend werden mehrfach neue Plugins in Jenkins genutzt, die Installation kann auf zwei Wegen[1] realisiert werden. Entweder über ein Interface der Jenkins-Installation oder händisch durch das Kopieren der Plugin-Datei in das [JENKINS_HOME/plugins] Verzeichnis. Für die hier verwendeten Erweiterungen wurde das vorhandene Interface genutzt. Regelmäßige Updates sind auf diese Wege ebenso einfach möglich, wie die Übersicht über die installierten Erweiterungen.

Das Interface lässt sich über [Manage Jenkins]>[Manage Plugins] erreichen. Unter [Available] findet man alle zur Verfügung stehenden Plugins. Die Konfiguration hängt anschließend von der Art der Erweiterung ab und wird im Zusammenhang mit der Erwähnung beschrieben. Allgemein kann man sagen, dass alle Plugins sich über das Web-Interface konfigurieren lassen und keinen Eingriff auf die Dateistruktur benötigen. Die Möglichkeit, eine automatische Einrichtung vorzunehmen, wird im Ausblick dieser Arbeit beschrieben.

[1] https://wiki.jenkins-ci.org/display/JENKINS/Plugins (20.11.2013)

5.2.2 Versionierung

Da Jenkins eine Vielzahl von Software zur Versionierung im Grundumfang oder durch Plugins unterstützt, ist an dieser Stelle in erster Linie die Frage zu beantworten, womit können Studierende am leichtesten Umgehen und was ist zukunftssicher und wird weitläufig eingesetzt. Zum Umfang von Jenkins gehört die Unterstützung von CVS und Subversion, das veraltete CVS wird dabei bei der weiteren Betrachtung außer acht gelassen. Sollte es aber, begründet durch alte Projekten genutzt werden, ist der Support seitens des CI-Servers möglich. Alternativ sollte über einen Wechsel nachgedacht werden. Dabei ist das Werkzeug cvs2svn[2] und eine Beschreibung der Befehle im direkten Vergleich[3] hilfreich.

Es ist sich also auf die verbreiteten Systeme Subversion [Web04] und Git [Web08] zu konzentrieren. Da beide Varianten Anhänger haben und es von Projekt zu Projekt vorkommen kann, dass das eine oder andere gewünscht wird, stellt das CI-System den Support für beide sicher.

Wie bereits erwähnt, lässt sich Subversion ohne weitere Plugins nutzen. Unter [Manage Jenkins]>[Configure Jenkins] lässt sich die gewünscht Subversion-Version einstellen und auch automatisch auf dem System installieren. In der Projektansicht unter [Configure] wählt man anschließend die gewünschte Versionierung aus und gibt die URL des Repositories an. Eine weitere zentrale Funktion ist das Polling des Repositories in einem festgelegten Intervall, ob Änderungen vorliegen. Dazu wählt man unter [Build Triggers]>[Poll SCM] aus und legt einen Zeitplan fest. Man kann entweder vorgeben wie oft ein Polling erfolgen soll oder einen konkreten Zeitpunkt. Im Beispiel (Abb. 5.5) bedeutet dies, dass alle 10 Minuten die Versionierung auf Änderungen überprüft wird. Durch den ‘H’-Trigger kann man die Last verteilen, so wird nicht immer genau alle 10 Minuten ein Durchlauf gestartet, sondern Rücksicht auf die Gesamtlast des Systems genommen.

Für die Nutzung von Git sind zwei Plugins notwendig, das Git Plugin[4] und das Git Client Plugin[5]. Unter [Manage Jenkis]>[Configure Jenkins] muss man beim Punkt [Git] den Pfad der Installation auf dem Server angeben, damit anschließen von der Jenkins-Installation die Befehle genutzt werden können. Anschließend ist die Funktion wie auch beim Subversion-Plugin beim Projekt einzurichten und die URL zum Repository anzugeben.

[2] http://cvs2svn.tigris.org/cvs2svn.html (20.11.2013)
[3] http://svn.apache.org/repos/asf/subversion/trunk/doc/user/cvs-crossover-guide.html (20.11.2013)
[4] https://wiki.jenkins-ci.org/display/JENKINS/Git+Plugin (20.11.2013)
[5] https://wiki.jenkins-ci.org/display/JENKINS/Git+Client+Plugin (20.11.2013)

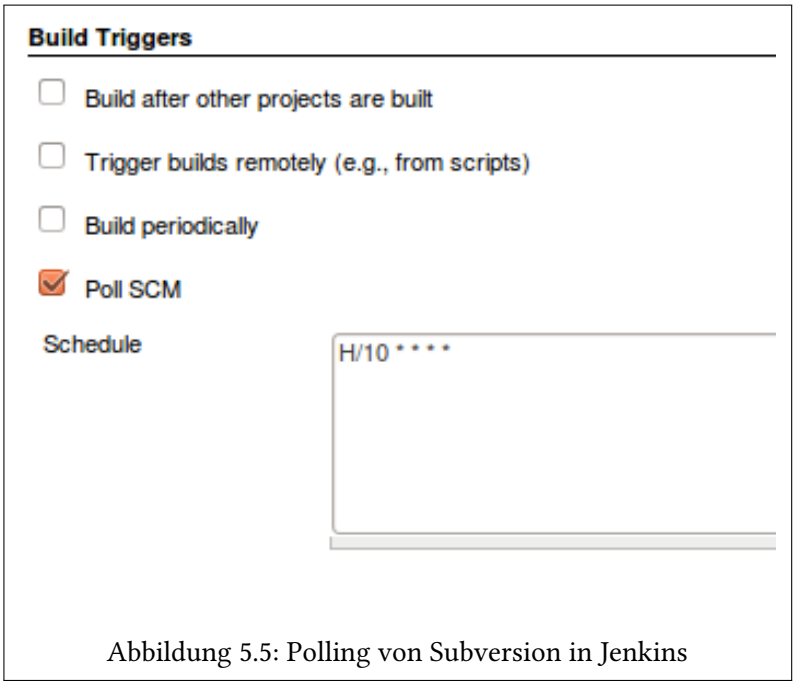


Abbildung 5.5: Polling von Subversion in Jenkins

Beide Versionierungen werden zusätzlich auf dem Server, der als gesamte CI-Plattform dient, installiert und können dann genutzt werden.

5.2.3 Serverumgebung

Bei der Wahl der Serverumgebung auf die Jenkins installiert wird, ist wieder der Fokus darauf zu legen, was einfach zu warten ist und unterstützt wird. In Rücksprache wurde sich dabei für eine Ubuntu-Server[6] Installation entschieden, welche in einer virtuellen Serverumgebung von VMWare-Workstation läuft. Unter Ubuntu gestaltet sich die Installation aller nötigen Pakete einfach und ist auf dem Jenkins-Wiki[7] ausführlich beschrieben.

Der Support durch die Hochschule ist für eine Einrichtung dieser Umgebung gegeben und durch den virtuellen Server lassen sich leicht weitere Installationen von Jenkins realisieren. Alle Abhängigkeiten lassen sich ebenfalls problemlos auf einem Ubuntu-Server installieren.

[6] http://www.ubuntu.com/download/server (20.11.2013)
[7] https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+on+Ubuntu (20.11.2013)

5.3 Jenkins-Plugins

In diesem Abschnitt werden alle weiteren Plugins von Jenkins beschrieben, die nötig sind, den Funktionsumfang zu erweitern. Um den Anforderungen zu entsprechen sind eine Reihe von Anpassungen nötig, wobei die Unterstützung durch von der Community bereitgestellten Plugins hergestellt wird. Wie auch Jenkins selber, sind die verwendeten Plugins frei verfügbar und in der Regel sogar frei anpassbar. Eine Übersicht der Plugins ist im Wiki von Jenkins zu finden. [8]. Auf den Detailseiten der einzelnen Erweiterungen findet man die Entwicker, das Repository und Statistiken zur Anzahl der Installationen.

5.3.1 Programmiersprachen

Um die Unterstützung für weitere Programmiersprachen herzustellen und dabei den gleichen Funktionsumfang, wie auch für Java, zu bieten, sind mehrere Plugins je Programmiersprache notwendig. Es wird auf die einzelnen Bereiche eingegangen und dargestellt, welche Erweiterung die Funktion für welche Programmiersprache herstellt.

5.3.1.1 Build-Management

Durch das Build-Management wird das automatische Bauen der Projekte auf der CI-Plattform realisiert. Dabei ist zu beachten, dass der Benutzer natürlich das passende Script (zum Beispiel Maven-Script) mitliefern muss, damit das System damit umgehen kann. Hier bedarf es einiger Schulung der Entwickler, ist das automatisierte Bauen aber erstmal in den normalen Entwicklungsprozess eingeliedert, nimmt es viel Arbeit ab und löst Probleme die ansonsten durch unsauberes Bauen auftreten könnten.

Java

Java wird durch die Standardinstallation unterstützt. Ein Ant-Plugin[9] sowie Maven-Plugin[10] bieten die Möglichkeit Java-Projekte zu bauen.

Einen Einstieg in Maven bietet folgende Seite: http://maven.apache.org/guides/getting-started/ (20.11.2013)
Um Ant zu erlernen besuchen sie folgende Seite: http://ant.apache.org/manual/index.html

[8] https://wiki.jenkins-ci.org/display/JENKINS/Plugins (20.11.2013)
[9] https://wiki.jenkins-ci.org/display/JENKINS/Ant+Plugin (20.11.2013)
[10] https://wiki.jenkins-ci.org/display/JENKINS/Maven+Project+Plugin (20.11.2013)

(20.11.2013)

C und C++

Natürlich lässt sich die Umsetzung automatisierter Aufgaben auch bei C und C++ ebenfalls mit Ant realisieren[11]. Leichter geht diese Umsetzung aber mit CMake[12] (cross-platform make). Auf der CI-Plattform muss dazu GCC installiert sein, über ein CMake-Script lässt sich dieses dann aufrufen. In Jenkins wird CMake durch ein Plugin[13] integriert.

C#

C# kann über ein Plugin ermöglicht werden, welches das Bauen von Visual Studio Projekten (.proj, .sln) übernimmt. [14] Da Jenkins auf einem Linux-System läuft, kann natürlich nicht direkt auf MSBuild und die nötigen Bibliotheken zugegriffen werden, da dafür eine Windows Installation notwendig ist. Eine Möglichkeit wäre, einen Slave einzurichten, also eine Jenkins-Installation in einer Windows Umgebung, die die Aufgaben für C# übernimmt. Eine Alternative mit Hilfe von Mono [Web09] wird vom Autor, Andre Broers, in einem Beitrag beschrieben [Broers (2013)]. Er hat sich genau dieser Problematik angenommen und eine Lösung entwickelt, die für die Jenkins-Installation übernommen werden kann.

5.3.1.2 Testen

Java

Zum Grundumfang von Jenkins gehören Unit Tests, das Modul dazu ist rund um JUnit aufgebaut und bietet damit die optimale Möglichkeit Java-Projekte zu testen.[15]

C und C++

Für C und C++ ist es notwendig eine auf Unit Tests basierte Lösung zu verwenden, damit Jenkins anschließend mit dem Resultat umgehen kann. Bei C ist CUnit[16] zu verwenden, C++

[11] http://ant.apache.org/ (20.11.2013)
[12] http://www.cmake.org/ (20.11.2013)
[13] https://wiki.jenkins-ci.org/display/JENKINS/cmakebuilder+Plugin (20.11.2013)
[14] https://wiki.jenkins-ci.org/display/JENKINS/MSBuild+Plugin 20.11.2013)
[15] https://wiki.jenkins-ci.org/display/JENKINS/Unit+Test (20.11.2013)
[16] http://cunit.sourceforge.net/ (20.11.2013)

wird durch CppUnit[17] erweitert. Um die Testergebnisse in Jenkins zu verwenden, gibt es mittlerweile das allgemeine Plugin xUnit[18], welches mit einer Vielzahl an Unit Test-Frameworks umzugehen weiß.

C#

Wie auch bei C++ muss ein Unit Test Framework genutzt werden. In diesem Fall ist es NUnit [19]. In Jenkins wird dann ebenso das xUnit-Plugin genutzt.

5.3.2 GUI und Layout

Die GUI-Funktionen werden ebenfalls durch mehrere Plugins angepasst. Um ein eigenes Layout zu ermöglichen, ohne an den Jenkins-Dateien direkt zu arbeiten, wird das Simple Theme Plugin[20] verwendet. Es ermöglicht den Einsatz einer eigenen CSS-Datei, welche die Standardelemente überschreibt.

Mit dem Dashboard-View Plugin[21] lassen sich die Elemente von Jenkins anordnen und darstellen. So wird den verschiedenen Benutzern eine verbesserte und angepasste Übersichtsseite präsentiert.

5.3.3 Benutzerverwaltung

Im nächsten Abschnitt wird im Detail auf die Benutzerverwaltung eingegangen. Das Role Strategy Plugin[22] ermöglicht dabei die vereinfachte Verwaltung.

5.3.4 Sonstige Erweiterungen

Eine andere Erweiterung in Kürze ist das Green Balls Plugin[23], um anstatt Blau dann Grün als Farbe eines erfolgreichen Vorgangs zu nutzen.

[17] http://sourceforge.net/projects/cppunit/ (20.11.2013)
[18] https://wiki.jenkins-ci.org/display/JENKINS/xUnit+Plugin (20.11.2013)
[19] http://www.nunit.org/ (20.11.2013)
[20] https://wiki.jenkins-ci.org/display/JENKINS/Simple+Theme+Plugin (20.11.2013)
[21] https://wiki.jenkins-ci.org/display/JENKINS/Dashboard+View (20.11.2013)
[22] https://wiki.jenkins-ci.org/display/JENKINS/Role+Strategy+Plugin (20.11.2013)
[23] https://wiki.jenkins-ci.org/display/JENKINS/Green+Balls (20.11.2013)

5.4 Umsetzung weiterer Anforderungen

Die Umsetzung weiterer Anforderungen geschieht durch die von Jenkins bereitgestellten Mittel.

5.4.1 Projekte

Projekte entsprechen Jobs in der Jenkins-Systematik. Die Projekt-Instanzen werden durch den Mechanismus der Job-Kopie erstellt. Es werden alle Einstellungen (Versionierung, Build-Management, Testing) übernommen und ein eigener Job angelegt. Durch das Anpassen des Repositorien kann ein Team an einem konkreten Job/ Projekt arbeiten. Durch das gezielte Ausblenden nicht zur Verfügung stehender Jobs hat nur das passende Team den Zugriff auf den Job und die Ergebnisse. Da Views nur von Lehrbeauftragten erstellt werden können, ist es Studierenden nicht möglich, ihnen nicht zustehende Jobs und Ergebnisse einzusehen.

5.4.2 Deployment

Das Deployment, verschieben der Artefakte auf einen lokalen oder entfernten Ort, wird durch einen Build-Schritt der Jobs von Jenkins realisiert. Dazu konfiguriert man eine Post-build Action, die nur nach einem erfolgreichen Build ausgeführt wird. Es ist an dieser Stelle möglich ein Shell-Script auszuführen und dadurch die entstandenen Artefakte an einen anderen Ort zu kopieren.

5.4.3 Projektgruppe

Alle Benutzer die an einem Projekt arbeiten, bekommen die passende View zugewiesen, die ihnen den Zugriff auf ihr konkretes Projekt ermöglicht. So kann vorerst einfach realisiert werden, dass nur bestimmte Benutzer und Benutzergruppen an einem Projekt arbeiten.

5.5 Benutzerverwaltung

Aus den in der Analysephase festgelegten Rollen für eine CI-Plattform an einer Hochschule, folgt die Umsetzung dieses Konzepts in Jenkins. Als erstes wird ein weiteres Mal die Rollenbeschreibung aufgegriffen und konkretisiert auf den Prototypen angewandt. Es ergibt sich durch Abhängigkeiten und mögliche Einschränkungen eine komplexere Struktur, die einer erneuten Abhandlung bedarf. Anschließend werden Beispiele für unterschiedliche Frontends gezeigt,

die, durch Modifikation von Jenkins, für die unterschiedlichen Aufgabenbereiche bereitgestellt werden können. Der Prototyp arbeitet ebenfalls mit diesen Frontends. Im Abschluss wird auf die Authentifizierung und Autorisierung der Benutzer in Jenkins eingegangen und erläutert, wie das System damit umgeht.

5.5.1 Rollenkonzept

Die Rolle, die mit den wenigsten Rechten ausgestattet ist, ist die des Studenten (Akteur: Student) ohne direkten Bezug zur Informatik. Dies soll alle Personen umfassen, die zwar Interesse an den bereitgestellten Artefakten und möglicherweise dem erfolgreichen Entwicklungsprozess haben, aber weiter keine Erfahrungen in der Softwareentwicklung oder administratives Interesse am konkreten Softwareprojekt haben. Für die CI-Plattform bedeutet dies, dass der Benutzer sich am System anmeldet und ausschließlich lesenden Zugriff auf die Projekte hat. Eine vorkonfigurierte Ansicht, die automatisch einem Benutzer mit dieser Rolle zugewiesen wird, unterstützt dabei die relevanten Bereiche zu finden.

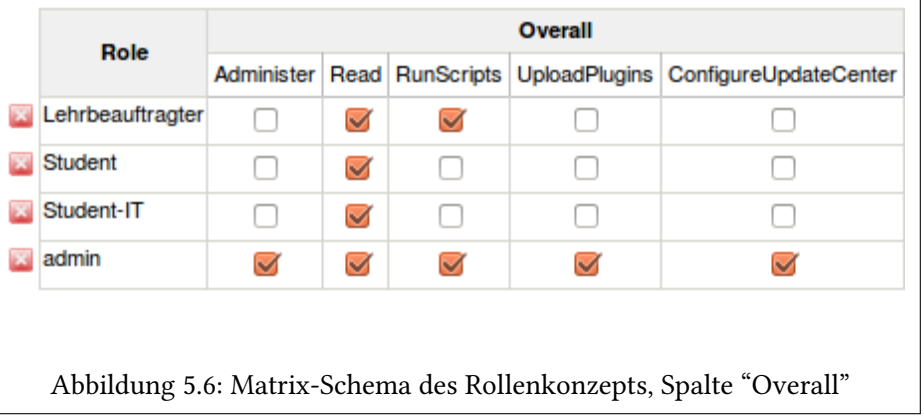
Mit umfassenderen, projektbezogenen Rechten sind die Informatik-Studenten (Akteur: Student-IT) ausgestattet. Diese haben zwar keine administrativen Kapazitäten, können aber die Prozesse anstoßen, die für das bauen und testen von Projekten zuständig sind. Durch den ebenfalls angelegten Zugriff auf die Versionierung, ist es diesen Personen möglich, Quellcode einzuchecken und zu ändern. Innerhalb des System, kann auch erst diese Rolle den eigentlichen Quellcode einsehen.

Mit den meisten Rechten, nach dem System-Administrator, sind die Lehrbeauftragten (Akteur: Lehrbeauftragter) ausgestattet. Ihre Rechte gestatten es Ihnen, andere Benutzer zu verwalten und weitere Ansichten (zum Beispiel auf bestimmte Projekte) für Benutzer anzulegen.

In Jenkins selber werden diese Rollen durch globale und projektbezogene Rollen umgesetzt. Wie auch bei der regulären Benutzerverwaltung, wird eine Rechte-Matrix verwendet, um die einzelnen Zuständigkeiten zuzuweisen. Da man es im allgemeinen Einsatz nur mit den festen Rollen zu tun haben wird, ist ein, den eigentlichen Benutzern übergeordnetes Rollenschema, von Vorteil bei der Administration. Anhand der verwendeten Begriffe in Jenkins wird im Folgenden die Umsetzung erläutert.

In der ersten Spalte (Abb. 5.6) bekommen alle Rollen einen lesenden Zugriff gestattet. Nur die Lehrbeauftragen dürfen Skripte ausführen (komplexere Jobs auf dem Jenkins-System). Die

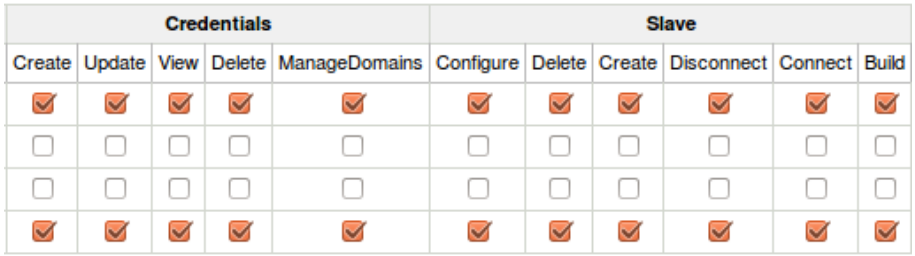
Administration bleibt alleine dem admin-Account vorbehalten.



Role	Overall				
	Administer	Read	RunScripts	UploadPlugins	ConfigureUpdateCenter
Lehrbeauftragter	☐	☑	☑	☐	☐
Student	☐	☑	☐	☐	☐
Student-IT	☐	☑	☐	☐	☐
admin	☑	☑	☑	☑	☑

Abbildung 5.6: Matrix-Schema des Rollenkonzepts, Spalte “Overall”

In der zweiten und dritten Spalte (Abb. 5.7) haben ebenfalls nur der Lehrbeauftragte und der Administrator die nötigen Rechte um Änderungen an den Benutzern vorzunehmen. Ferner bleibt es diesen Benutzergruppen vorbehalten Slave-Systeme zu verwalten, auch wenn dies vorerst nicht nötig ist.



Credentials					Slave					
Create	Update	View	Delete	ManageDomains	Configure	Delete	Create	Disconnect	Connect	Build
☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑

Abbildung 5.7: Matrix-Schema des Rollenkonzepts, Spalten “Credentials” und “Slave”

In weiteren Spalten (Abb. 5.8) haben alle Benutzerrollen die Rechte Jobs (Projekte) einzusehen und lesenden Zugriff zu erhalten. Nur Student-IT und Lehrbeauftragter darf diese auch konfigurieren. Das Erstellen von Views (Ansichten auf das System, welche Elemente, Projekte werden dargestellt) ist wieder nur den Lehrbeauftragten vorbehalten.
Testweise eingerichtete Benutzer (Abb. 5.9) bekommen dann, ebenfalls in einer Benutzermatrix, diese Rollen zugeordnet. Dieser Vorgang nimmt einiges an Organisationsaufwand ab, da man nicht jedem Benutzer manuell die nötigen Rechte vergeben muss. Auch entstehen so keine

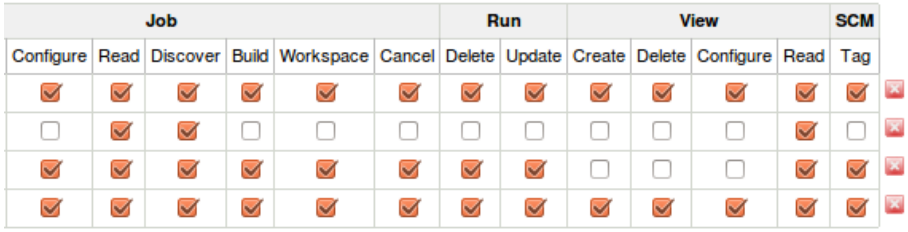


Abbildung 5.8: Matrix-Schema des Rollenkonzepts, Spalten “Job”, “Run”, “View” und “SCM”

Fehler bei der Rechtevergabe.

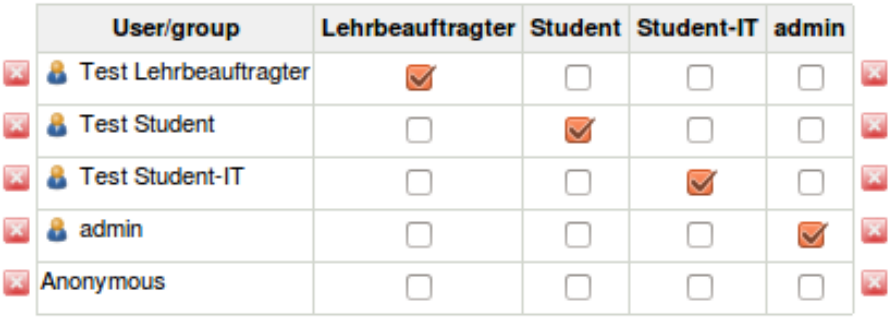


Abbildung 5.9: Benutzerverwaltung

5.5.2 Frontend von Jenkins

Die Webanwendung von Jenkins bietet umfangreiche Möglichkeiten, die darzustellenden Informationen den individuellen Bedürfnissen anzupassen. Folgende Abbildungen zeigen beispielhaft den angestrebten Informationsumfang der einzelnen Dashboards für die Benutzerrollen. Zur Veranschaulichung wurden zwei Projekte angelegt, wobei das eine Projekt fehlschlägt und das andere Projekt erfolgreich gebaut wird. Wie bereits in 5.3.2 beschrieben, wurde das Plugin “Jenkins Dashboard View” benutzt.

(Abb. 5.10) ist die einfachste Darstellung und wird der Benutzerrolle “Student” als Default-View zugewiesen. Fokus der verwendeten Elemente liegt auf einer kompakten Darstellung der ver-

folgten Projekte, einer Übersicht über die Build und zuletzt durchgeführten Build. Der Anteil der erfolgreich durchgeführten Tests wird in einer einfachen Darstellung als Diagramm gezeigt.

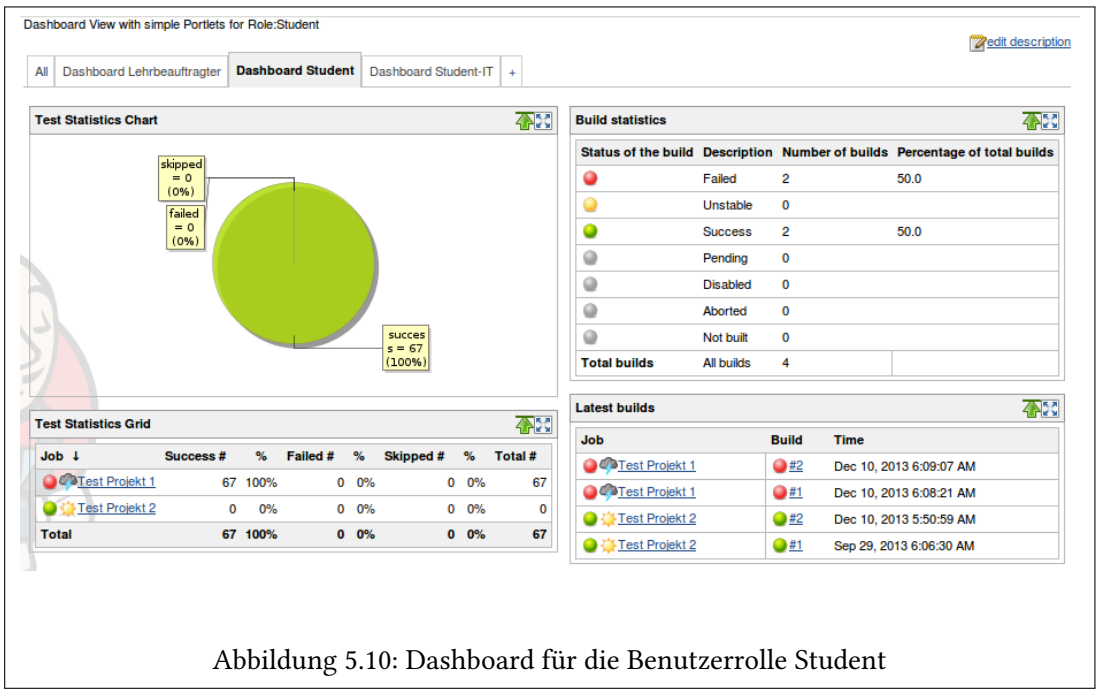


Abbildung 5.10: Dashboard für die Benutzerrolle Student

(Abb. 5.11) und (Abb. 5.12) ist die Default-View der Benutzerrolle “Student-IT”. Die Projekte/ Jobs in Jenkins werden prominenter dargestellt und zusätzliche Fehlermeldungen und Teststatistiken eingebunden. Compiler-Warnungen, Checkstyle und eine Übersicht über instabile Jobs sind besonders für “Student-IT” interessant.

(Abb. 5.13) stellt die Default-View für den “Lehrbeauftragten” dar. Hier ist besonders ein schneller Überblick über alle Jobs und Projekte wichtig und weniger die konkreten Meldungen einzelner Jobs. Es werden mehr Informationen als bei der Rolle “Student” dargestellt.
Alle Abbildungen sind nur ein Beispiel für die umfangreichen Möglichkeiten Jenkins anzupassen. Weitere Portlets lassen sich entweder über zusätzliche Plugins einbinden oder gar selbst erstellen. Die Dashboard-View ist dabei so modular aufgebaut, dass ohne weiteres andere Elemente unterstützt werden. Das Layout und Farbgebung der Anwendung lässt sich über CSS anderen Portalen anpassen, um den Wiedererkennungswert zu erhöhen.

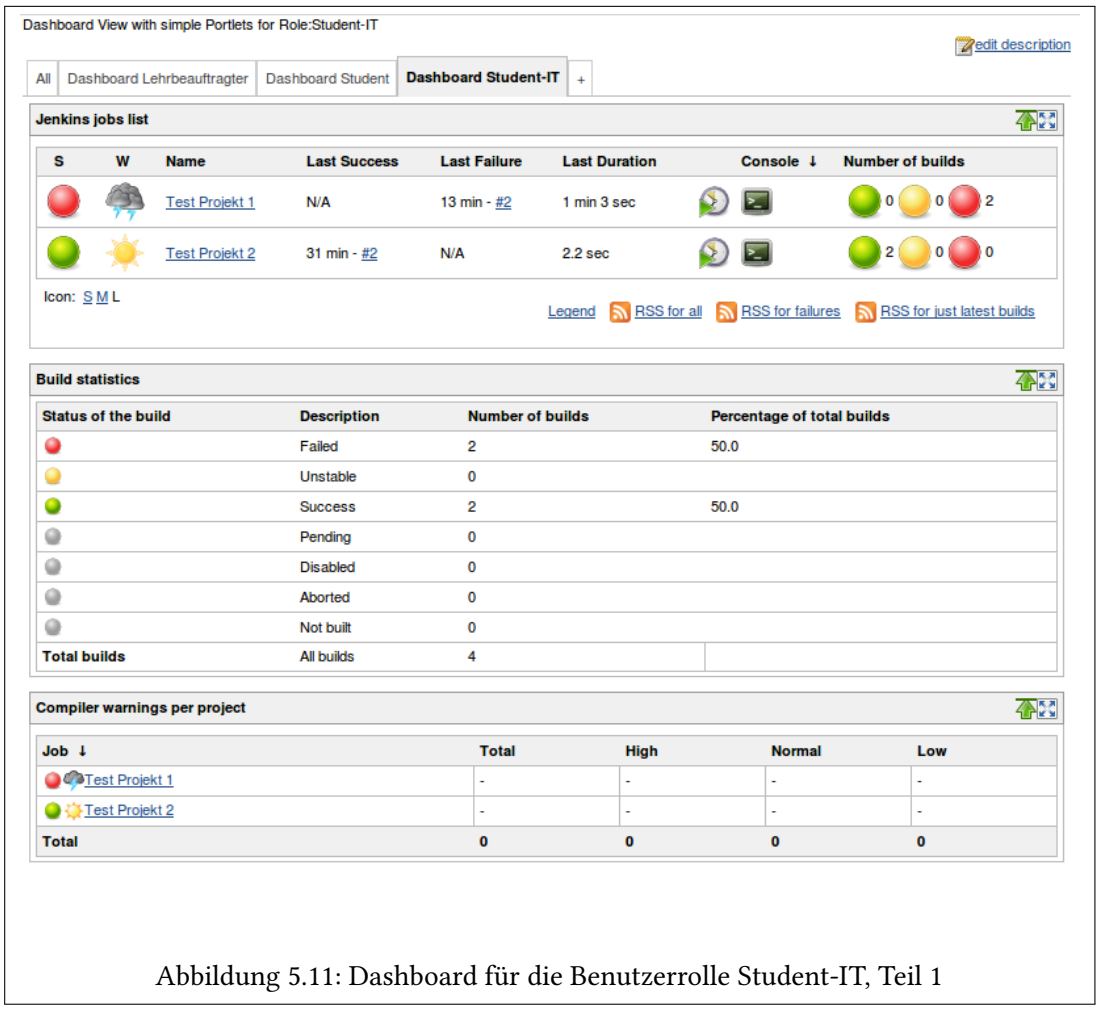


Abbildung 5.11: Dashboard für die Benutzerrolle Student-IT, Teil 1

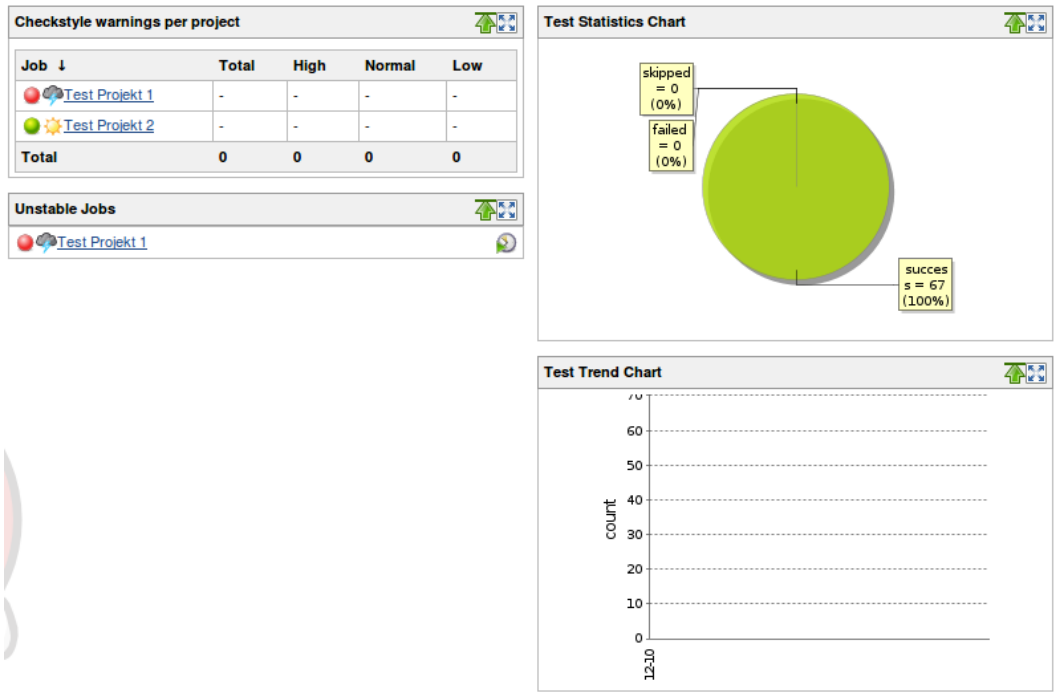


Abbildung 5.12: Dashboard für die Benutzerrolle Student-IT, Teil 2

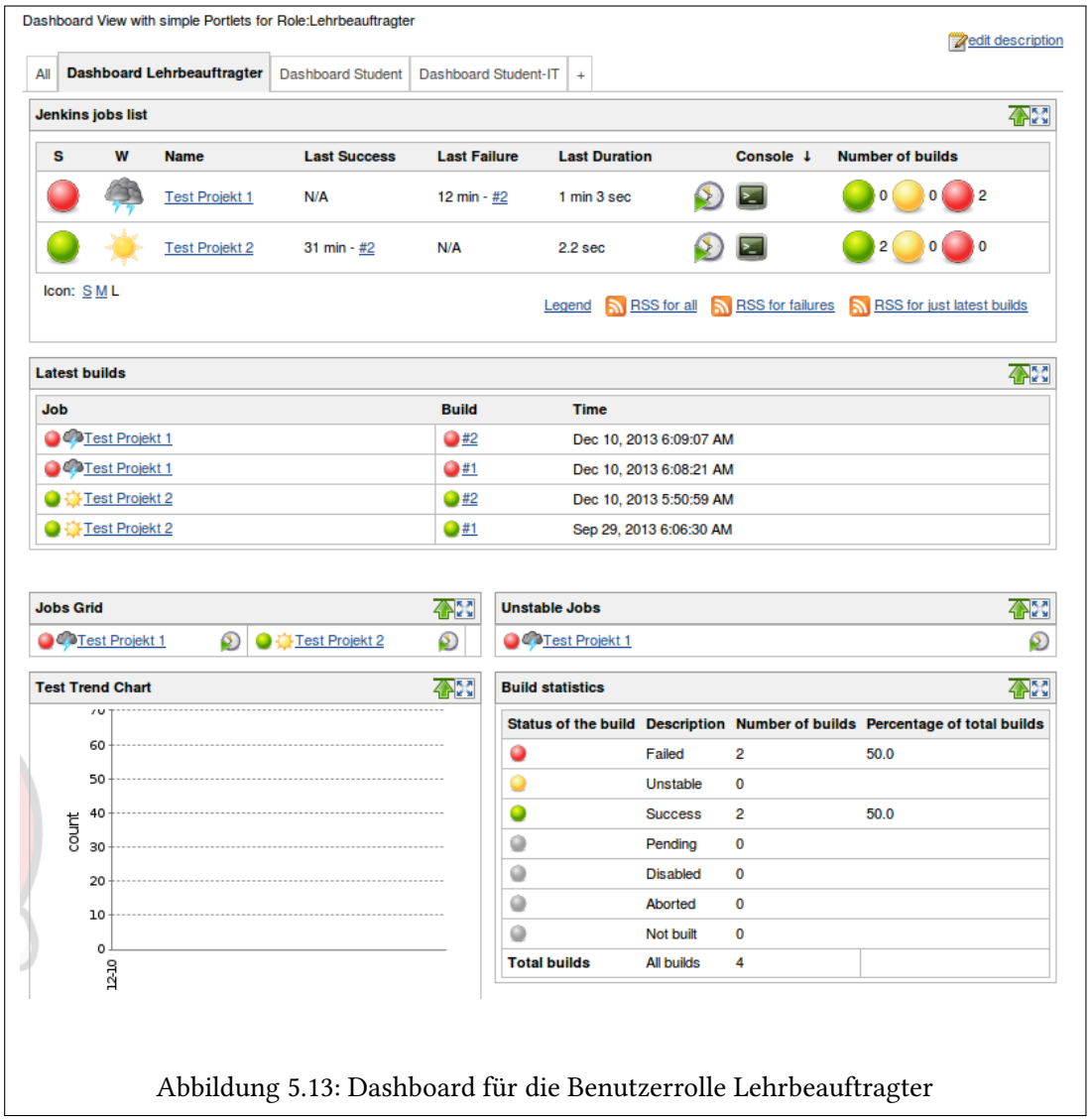


Abbildung 5.13: Dashboard für die Benutzerrolle Lehrbeauftragter

5.5.3 Authentifizierung und Autorisierung

Es gibt zwei Hauptachsen bei Jenkins[24], welche den Zugriff von Benutzer auf das System kontrollieren. Die erste Ebene, mit denen Benutzer in Kontakt kommen ist der "Security Realm". Bei der Anmeldung wird der Benutzername und das Passwort überprüft und anschließend sichergestellt, zu welcher Gruppe der Benutzer gehört.

Die zweite Achse wurde bereits bei der Benutzerverwaltung beschrieben. Die Autorisations-Strategie überprüft, welcher Benutzer Zugriff auf welche Inhalte und Möglichkeiten hat. Zum Einsatz kommt die bereits beschriebene rollenbasierte Matrix.

Im Menüpunkt [Manage Jenkins]>[Configure Global Security] lassen sich sowohl der "Security Realm" als auch die "Authorization" auswählen. Die Einrichtung erfolgt dann in getrennten Menüs. Eine Beispielhafte Konfiguration (Abb. 5.14) ist die verwendet der Jenkins eigenen Benutzerdatenbank, dabei kann man festlegen ob sich neue Benutzer registrieren dürfen, oder nur eingerichtete Benutzer Zugriff erhalten.

Es gibt auch weitere fortgeschrittene Möglichkeiten die Authentifizierung und Autorisierung sicherzustellen. Eine weiterführende Idee auf dieser Basis wird im Ausblick beschrieben.

[24] https://wiki.jenkins-ci.org/display/JENKINS/Securing+Jenkins(10.12.2013)

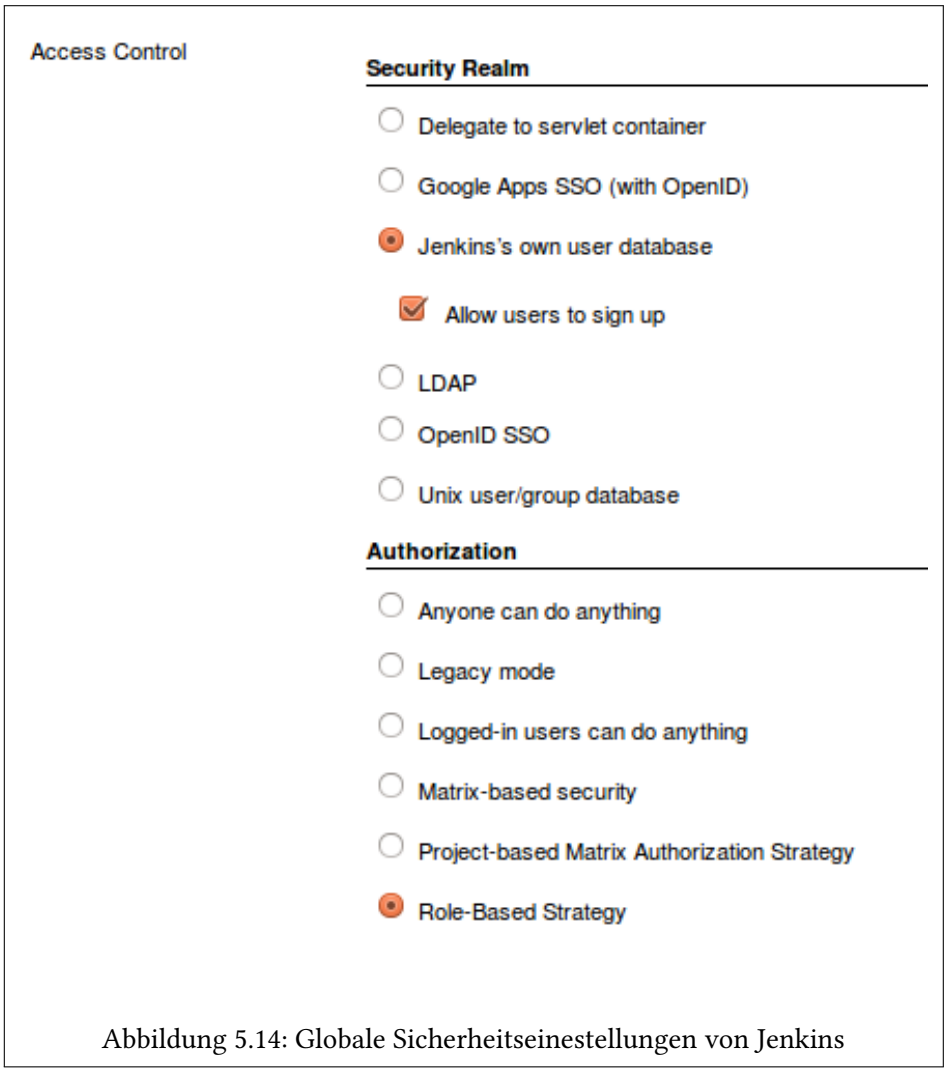


Abbildung 5.14: Globale Sicherheitseinestellungen von Jenkins

6 Zusammenfassung und Ausblick

6.1 Zusammenfassung

Die interdisziplinäre Zusammenarbeit von Studierenden der Informatik und anderen Fachrichtungen an Hochschulen steht vor Problemen, da häufig zu Beginn des Studiums zwar grundlegende Programmiertechniken vermittelt werden, aber Werkzeuge für die Kooperation, Testing und Code-Delivery zu kurz kommen. In dieser Arbeit wurde sich des Problems angenommen, indem die Technik der Continuous Integration mit Hinblick auf diese Problematik hin untersucht und ein Prototyp beschrieben wurde.

Es wurde ein Überblick verschafft wie Continuous Integration entstanden ist, was die einzelnen Schritte dabei sind und welche Techniken zum Einsatz kommen. Der Zusammenhang von CI und agiler Softwareentwicklung wird dargelegt. Die CI-Kette spielt eine wichtige Rolle für die Schritte, die im Anschluss an die Analyse-Phase bei der Erstellung des Prototypen relevant waren. Bei der Analyse der Anforderungen wurde eng mit Frau Prof. Dr. Wendholt zusammengearbeitet, um die nötigen Kernkomponenten für ein solches System festzulegen. Die Auswahl der Software fand unter diesen Gesichtspunkten statt. Der dann ausgewählte Kandidat für die CI-Plattform “Jenkins” war die beste Möglichkeit, schnell einen lauffähigen und für den konkreten Bereich einsatzfähigen Prototypen, zu erlangen.

Bei der daraus resultierenden Serverumgebung wurde für alle Komponenten Wert auf offene Lizenzen gelegt, die beim Betrieb an einer Hochschule keine rechtlichen Probleme verursachen werden. Die verwendeten Module von Jenkins sind beispielhaft für die hervorragende Erweiterbarkeit der Software. Besonders wichtig im Umgang mit den Daten der Studierenden war ein sauber ausgearbeitetes Rollenkonzept und das Verfahren, welches für die Authentifizierung und Autorisierung am System verwendet wurde. Dabei ist zu beachten, dass das verwendete Verfahren nur eine der Möglichkeiten ist, die Aufgrund der knappen Zeit genutzt wurde. Im Ausblick folgen weitere Ansätze und Erleichterungen für die Lehrbeauftragen, die ohne weiteres mit den gegebenen Mitteln und ausreichend Zeit möglich sind.

Der Prototyp ist lauffähig und kann für Softwareprojekte mit unterschiedlichen Programmiersprachen genutzt werden. Da der Einsatz von Jenkins vielfach bewährt ist, wurde auf einen Usability-Test der Oberfläche vorerst verzichtet und sich auf wesentliche Bestandteile des Prototypen und eine saubere Beschreibung der Anwendungsfälle konzentriert. Sollte die Bewertung der Plattform positiv ausfallen, steht einer Weiterentwicklung auf dieser Basis nichts im Wege. Der Grundstein für ein sauber arbeitendes CI-System an Hochschulen ist somit gelegt.

6.2 Ausblick

Im abschließenden Kapitel wird relativ umfangreich auf die Weiterentwicklungsmöglichkeiten für die Software-Plattform eingegangen. Mit vielen Elementen wurde sich während der Entwicklungsphase umfassender beschäftigt, konnten aber aufgrund von Zeitmangel nicht umgesetzt werden. Um diese Arbeiten zu einem späteren Zeitpunkt zu erleichtern, wird auf die einzelnen Punkte eingegangen und ein erster Ansatz geliefert, wie diese umzusetzen sind.

6.2.1 Deployment der CI-Plattform

Der Protoyp wird momentan noch größtenteils manuell eingerichtet und bedarf der nachträglichen Einrichtung. Im aktuellen Zustand wird der komplette Funktionsumfang; Plugins, Rollenkonzept, Views; kopiert, die konkreten Einstellungen müssen aber nach wie vor vorgenommen werden. Viele Einstellungen, von Jenkins selber und von den verwendeten Plugins, sind aber in XML-Dateien abgelegt und können durch diese auch angepasst werden.

Als Beispiele für diese Art der Konfiguration sollen zum einen die “config.xml” eines Jobs dienen und die des Plugins “Role Strategy”.

```
<scm class="hudson.scm.SubversionSCM" plugin="subversion@1.51">
    <locations>
      <hudson.scm.SubversionSCM_-ModuleLocation>
        <remote>file:///home/strecki/testrepo</remote>
        <local>.</local>
        <depthOption>infinity</depthOption>
        <ignoreExternalsOption>false</ignoreExternalsOption>
      </hudson.scm.SubversionSCM_-ModuleLocation>
```

```
    </locations>
     <excludedRegions></excludedRegions>
    <includedRegions></includedRegions>
    <excludedUsers></excludedUsers>
    <excludedRevprop></excludedRevprop>
    <excludedCommitMessages></excludedCommitMessages>
    <workspaceUpdater class="hudson.scm.subversion.UpdateUpdater"/>
    <ignoreDirPropChanges>false</ignoreDirPropChanges>
    <filterChangelog>false</filterChangelog>
</scm>
```

In diesem Auschnitt aus der “config.xml” des Testjob ist beschrieben, dass Subversion als Versionierung verwendet wird und der Ort des Repositories. Dies lässt sich natürlich so auch außerhalb von Jenkins anpassen, so das für einen Job/ Projekt dann keine Schritte in Jenkins selber nötig sind. Beispielweise lassen sich diese Einstellungen für daraus abgeleitete Jobs/ Projekte auch übernehmen.

```
      <role name="Student" pattern=".*">
        <permissions>
          <permission>hudson.model.View.Read</permission>
          <permission>hudson.model.Hudson.Read</permission>
          <permission>hudson.model.Item.Discover</permission>
          <permission>hudson.model.Item.Read</permission>
        </permissions>
        <assignedSIDs>
          <sid>Test Student</sid>
        </assignedSIDs>
      </role>
```

In diesem Ausschnitt aus der Konfigurationsdatei von Jenkins ist das verwendete Rollenkonzept beschrieben und kann angepasst werden

An weiteren Stellen sind in dieser Form die anpassbaren Elemente von Jenkins aufgelistet und können durch umfangreiche Skripte dazu genutzt werden, eine Instanz nahezu automatisch einzurichten. Diese Möglichkeit ist ein sehr spannender Ausblick und wurde bislang in dieser Form auch noch nicht umgesetzt.

6.2.2 Usability-Tests der CI-Plattform

Da das System, bedingt durch das Build-Management und die Test-Systeme nicht ohne zu erwerbende Vorkenntnisse zu benutzen ist, sind für einen Einsatz Usability-Tests nicht wegzu-

denken. Dabei kann man diese zweistufig gestalten und im ersten Schritt die Benutzbarkeit der Oberfläche testen und anschließend, als fortgeschrittenen Tests, die Fähigkeiten von Benutzern selber das Build-Management und Testing für unterschiedliche Programmiersprachen zu nutzen.

Die Usability-Tests des GUI werden für jede Benutzergruppe getrennt stattfinden, da man mit abweichenden Vorkenntnissen zu rechnen hat. Dabei werden einfache Aufgaben gestellt, aber ansonsten alle nötigen Projekte, Jobs und Dateien vorgeben. Es wird ausschließlich getestet, wie schnell die Benutzer mit den Elementen zurecht kommen und die gewünschten Einstellungen finden.

Die weiterführenden Tests richten sich in erster Linie an die Benutzergruppen Student-IT und Lehrbeauftragter und sollen testen, wie typische Aufgaben im produktiven Einsatz zu bewältigen sind. Dies kann zum Beispiel sein, zu einem Projekt das nötige Build-Script einzurichten und die Tests durchführen zu lassen.

Zusätzlich werden im Rahmen von Usability-Tests auch Handbücher angefertigt, die nicht nur in die Arbeitsweise von Continuous Integration speziell für diesen Anwendungskontext einführen, sondern auch die Benutzung des Programmes erläutern.

6.2.3 Weitere Programmiersprachen

Um weitere Programmiersprachen hinzuzufügen sind, wie in Kapitel 5 beschrieben mehrere Elemente nötig. Es ist durch Ant oder ein vergleichbares Werkzeug und dem dazugehörigen Jenkins-Plugins sicherzustellen, dass automatische Builds durchgeführt werden können. Weiter muss es ein Unit-Test-Framework für diese Sprache geben, damit Jenkins die Ergebnisse interpretieren kann. Diese Möglichkeit ist in der Regel durch das xUnit-Plugin abgedeckt. Somit ist die Grundfunktion erfüllt, Code-Metriken sind natürlich eine hilfreiche Funktion, aber nicht zwingend erforderlich. Die verwendete Versionierung hat keine Probleme mit den erwartbaren Programmiersprachen.

6.2.4 Erweiterte Benutzerauthentifizierung und Autorisierung im Hochschulkontext

In der aktuellen Plattform müssen die Benutzer noch manuell angelegt werden und bekommen eine Rolle zugewiesen, über die dann anschließend die Autorisierung beim System durchgeführt wird. Eine andere Lösung und besonders für die Lehrbeauftragten von Vorteil ist es, wenn

die Benutzer für eine Veranstaltung, und damit CI-Plattform, automatisch angelegt werden.

Dazu ist es denkbar, auf die verfügbaren Informationen der Hochschule zuzugreifen. In der Regel gibt es einen Benutzeraccount um die Dienste des Rechenzentrums und der Studierenden-Selbstverwaltung zu nutzen. Da diese Daten hochsensibel sind, kommt ein direkter Zugriff nicht in Frage und es ist ein Umweg notwendig. Um dennoch einen Zugriff zu realisieren, ist ein sicherer Verzeichnisdienst auf der Seite der Hochschule notwendig, auf den man zwar Zugriff hat, welcher aber nicht die Passwörter der Studierenden in irgendeiner Form einsehbar übermittelt. Da Jenkins LDAP[1] unterstützt, ist eine Lösung auf dieser Basis denkbar. LDAP würde als Verzeichnisdienst die Informationen vorhalten und automatisch aus dem Hochschulsystem befüllt werden. Die Anmeldung an einem konkreten Jenkins-System erfolgt dann auf Basis des Benutzernamens, des Semesters und der belegten Veranstaltung. Zur Authentifizierung und sicheren Übertragung der Informationen kommt zusätzlich zu LDAP ein Kerberos zum Einsatz. Der Dienst für den authentifiziert werden würde, ist in diesem Fall dann der Zugriff auf den Verzeichnisdienst und damit der Erhalt der nötigen Autorisierung für die CI-Plattform.

Hierzu sind natürlich einige Modifikationen an Jenkins selber erforderlich, dadurch das es ein Open-Source-System ist, steht diesen Änderungen aber nichts im Wege.

[1] Lightweight Directory Access Protocol

Literaturverzeichnis

[Web10] *Continuous Delivery vs Continuous Deployment.* http://continuousdelivery.com/2010/08/continuous-delivery-vs-continuous-deployment/. – [Online; abgerufen 15-Juli-2013]

[Web11] *Continuous Integration auf ThoughtWorks.* http://www.thoughtworks.com/de/continuous-integration. – [Online; abgerufen 15-Juli-2013]

[Web02] *CruiseControl Dokumentation.* http://cruisecontrol.sourceforge.net/overview.html. – [Online; abgerufen 15-Juli-2013]

[Web07] *GCC Manual.* http://gcc.gnu.org/onlinedocs/. – [Online; abgerufen 15-Juli-2013]

[Web06] *GCC Wiki.* http://gcc.gnu.org/wiki. – [Online; abgerufen 15-Juli-2013]

[Web08] *Git Dokumentation.* http://git-scm.com/documentation. – [Online; abgerufen 15-Juli-2013]

[Web03] *Jenkins Wiki.* https://wiki.jenkins-ci.org/display/JENKINS/Home. – [Online; abgerufen 15-Juli-2013]

[Web12] *Maven Dokumentation.* http://maven.apache.org/guides/index.html. – [Online; abgerufen 15-Juli-2013]

[Web09] *Mono Dokumentation.* http://www.mono-project.com/Start. – [Online; abgerufen 15-Juli-2013]

[Web04] *Subversion Dokumentation.* http://subversion.apache.org/docs/. – [Online; abgerufen 15-Juli-2013]

[Web05] *Subversion Wiki.* http://wiki.apache.org/subversion/. – [Online; abgerufen 15-Juli-2013]

[Web01] *Team Fundation Server Dokumentation.* http://msdn.microsoft.com/en-us/library/aa730884%28v=vs.80%29.aspx. – [Online; abgerufen 15-Juli-2013]

[Web13 2013] *ThoughtWorks CI Feature Matrix.* http://confluence.public.thoughtworks.org/display/CC/CI+Feature+Matrix. 2013. – [Online; abgerufen 10-Dezember-2013]

[Beck 2004] Beck, Kent: *Extreme Programming Explained: Embrace Change.* Addison-Wesley Longman, 2004

[Berg 2012] Berg, Alan M.: *Jenkins Continuous Integration Cookbook.* Packt Publishing, 2012

[Bjoern Feustel 2012] Bjoern Feustel, Steffen S.: *Continuous Integration in Zeiten agiler Programmierung.* http://www.heise.de/developer/artikel/Continuous-Integration-in-Zeiten-agiler-Programmierung-1427092.html. 2012. – [Online; abgerufen 15-Juli-2013]

[Broers 2013] Broers, Andre: *Create a Mono C# Buildserver using Jenkins and GitHub.* http://blog.bekijkhet.com/2013/01/create-mono-c-buildserver-using-jenkins.html?goback=.gde_4446702_member_203830293. 2013. – [Online; abgerufen 15-Juli-2013]

[C. Titus Brown] C. Titus Brown, Rosangela Canino-Koning: *The Architecture of Open Source Applications: Continuous Integration.* http://www.aosabook.org/en/integration.html. – [Online; abgerufen 15-Juli-2013]

[Cockburn 2000] Cockburn, Alistair: *Agile Software Development.* 2000

[Colin Walters 2013] Colin Walters, Daniel M. G.: The Future of Continuous Integration in GNOME. In: *IEEE* (2013)

[David Cohen 2003] David Cohen, Patricia C.: Agile Software Development. (2003)

[Dyer 2008] Dyer, Dan: *Why are you still not using Hudson?* http://blog.uncommons.org/2008/05/09/why-are-you-still-not-using-hudson/. 2008. – [Online; abgerufen 15-Juli-2013]

[Fazreil Amreen Abdul 2012] Fazreil Amreen Abdul, Menesly Cheah Siow F.: Implementing Continuous Integration towards Rapid Application Development. In: *ICIMTR2012* (2012)

[Fowler] Fowler, Martin: *Continuous Integration Artikel.* http://martinfowler.com/articles/continuousIntegration.html. – [Online; abgerufen 15-Juli-2013]

[Fowler 2006] Fowler, Martin: Continuous Integration. (2006)

[Groeschel 2012] Groeschel, Prof. Dr. M.: *Entscheidungsfaktoren zum Einsatz von Open-Source-Software an Hochschulen*. http://www.opensourcepublicsector.de/?p=85. 2012. – [Online; abgerufen 15-Juli-2013]

[Jesper Holck 2004] Jesper Holck, Niels J.: Continuous Integration and Quality Assurance: A Case Study of two Open Source Projects. In: *Australasian Journal of Information Systems* (2004)

[Jez Humble 2010] Jez Humble, David F.: *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley, 2010

[Jez Humble] Jez Humble, Rolf R.: *The Agile Maturity Model: Applied to Building and Releasing Software*. http://www.thoughtworks-studios.com/sites/default/files/resource/agile_maturity_model.pdf. – [Online; abgerufen 15-Juli-2013]

[Jon Bowyer 2006] Jon Bowyer, Janet H.: Assessing Undergraduate Experience of Continuous Integration and Test-Driven Development. (2006)

[Kluth 2008] Kluth, Sascha: *Einfuehrung eines Open Source Build Management und Continuous Integration Systems in eine Java-Entwicklungsumgebung*, HAW Hamburg, Bachelorarbeit, 2008. – URL http://edoc.sub.uni-hamburg.de/haw/volltexte/2008/468/pdf/Bachelorarbeit_Sascha_Kluth.pdf

[Loeliger 2009] Loeliger, Jon: *Version Control with Git*. O'REILLY, 2009

[Martin R. Bakal 2012] Martin R. Bakal, Paridhi V.: Continuous Integration in agile development. (2012)

[Mike Snell 2013] Mike Snell, Lars P.: *Microsoft Visual Studio 2012 unleashed*. Sams, 2013

[Paul M. Duvall 2007] Paul M. Duvall, Andrew G.: *Continuous Integration: Improving software quality and reducing risk*. Addison-Wesley, 2007

[Peter Abrahamsson 2009] Peter Abrahamsson, Xiaofeng W.: Lots done, more to do: the current state of agile systems development research. (2009)

[Popp 2006] Popp, Gunther: *Konfigurationsmanagement mit Subversion, Ant und Maven*. dpunkt.verlag, 2006

[Sam Guckenheimer 2013] Sam Guckenheimer, Neno L.: *Visual Studio Team Foundation Server 2012: Adopting Agile Software Practices, From Backlog to Continuous Feedback*. Addison-Wesley, 2013

[Vincent Massol 2005] Vincent Massol, Timothy O.: *Maven: A Developer's Notebook*. O'REILLY, 2005

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §(4) bzw. §(4) ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 12. Dezember 2013 Till Streckwaldt