Institut für Softwaretechnologie
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Fachstudie Nr. 122

Kollaborative Entwicklungsplattformen

Aleksander Tolev
Modood Alvi
Timotheus Giuliani

Studiengang: Softwaretechnik

Prüfer: Prof. Dr. Jochen Ludewig

Betreuer: Dipl.-Inf. Holger Röder
Dipl.-Inf. Markus Knauß

begonnen am: 15. Januar 2010

beendet am: 26. August 2010

CR-Klassifikation: D.2.6, K.6.1

Inhaltsverzeichnis

1. Einleitung **9**
1.1. Hintergrund . . . 9
1.2. Aufgabenstellung . . . 9
1.3. Leserkreis . . . 10
1.4. Vorgehen . . . 10
1.4.1. Verwandte Arbeiten . . . 11
1.4.2. Projektmerkmale . . . 11
1.4.3. Tätigkeiten . . . 12
1.4.4. Projektprofile . . . 12
1.4.5. Nutzungsszenarien . . . 12
1.4.6. Marktübersicht . . . 12
1.4.7. K.O.-Kriterien . . . 13
1.4.8. Vergleichskriterien . . . 13
1.4.9. Bewertungsschema . . . 14
1.4.10. Empfehlung . . . 14

2. Verwandte Arbeiten **15**
2.1. Vorgehensweise . . . 15
2.2. Schlüsselwörter . . . 16
2.3. Suchergebnisse . . . 16
2.3.1. Erkenntnisse aus den Suchergebnissen . . . 18

3. Projektprofile **19**
3.1. Definition Projekt . . . 19
3.2. Projektmerkmale von Software-Projekten . . . 19
3.3. Projektprofile nach Merkmalen . . . 21

4. Nutzungsszenarien **23**
4.1. Tätigkeiten . . . 23
4.1.1. Werkzeugtabelle . . . 26
4.2. Interviews . . . 27
4.2.1. Interviewdurchführung . . . 28
4.2.2. Interviewergebnisse . . . 30
4.3. Nutzungsszenarien . . . 30

5. Kollaborationsplattformen **33**
5.1. Definition . . . 33
5.2. Marktübersicht . . . 33
5.3. K.O.-Kriterien . . . 34
5.3.1. Anforderungen . . . 34
5.4. Kurzliste . . . 36
5.5. Vergleich . . . 37

6. Bewertung der Plattformen **45**
6.1. Bewertung . . . 45
6.1.1. Bewertungsschema . . . 46
6.1.2. Ein-Personen-Projekt . . . 48
6.1.3. Software-Praktikum . . . 48
6.1.4. Studienprojekt . . . 49
6.1.5. Internes-Projekt . . . 50
6.1.6. Usability und Integration . . . 51
6.1.7. Ergebnisse der Usability und Integration . . . 52
6.2. Empfehlung . . . 56
6.2.1. Empfehlung für liberAda . . . 57

7. Fazit **59**
7.1. Zusammenfassung . . . 59
7.2. Ergebnisse . . . 59
7.3. Schwierigkeiten der Fragestellung . . . 59
7.4. Ausblick . . . 60

A. Interview-Leitfaden **61**

B. Interview-Fragenkatalog **67**

C. Interview-Ergebnisse **77**
C.1. Ein-Personen-Projekt . . . 77
C.2. Softwarepraktikum . . . 79
C.3. Studienprojekt . . . 81
C.4. Internes-Projekt . . . 83

D. Plattformen **85**
D.1. Marktübersicht . . . 85
D.2. K.O-Kriterien . . . 92
D.3. Vorgehensmodell . . . 93

E. Bewertung **99**
E.1. Ein-Personen-Projekt . . . 100

E.2. Software-Praktikum 101
E.3. Studienprojekt 102
E.4. Internes-Projekt 103

Literaturverzeichnis **107**

Abbildungsverzeichnis

4.1. Taxonomie der Tätigkeiten 24

5.1. Anforderung an die Plattform 35

6.1. Screenshot GForge 53
6.2. Screenshot GForge Bugtracker 53
6.3. Screenshot JavaForge 55
6.4. Screenshot JavaForge Bugtracker 55

Tabellenverzeichnis

3.1. Projektmerkmale 20
3.2. Projektprofile nach Merkmalen 21
3.3. Repräsentanten für Interviews 22

4.1. Tätigkeiten im Softwareentwicklungsprozess 26
4.2. Werkzeugtabelle 27
4.3. Einstufung der Tätigkeiten 30
4.4. Ausgeführte Tätigkeiten in den Projektprofilen (Nutzungsszenario) 31

5.1. Kurzliste 37
5.2. Bewertung im Vergleich 38
5.3. Rudimentäre und volle Unterstützung der Tätigkeiten 40
5.4. Vergleich der Plattformen 41

6.1. Bewertungsschema 47
6.2. Beispiel der Bewertung 47
6.3. Rangfolge für das Ein-Personen-Projekt 48
6.4. Rangfolge für das Software-Projekt 49
6.5. Rangfolge für das Studienprojekt 50

6.6. Rangfolge für das Interne-Projekt 50
6.7. Bewertung der Usability und Integration bei GForge 54
6.8. Bewertung der Usability und Integration bei JavaForge 56
6.9. Empfehlungen für die Projektklassen 57
6.10. Konkrete Bewertung für liberAda 57

D.1. Marktübersicht Teil 1 86
D.2. Marktübersicht Teil 2 87
D.3. Marktübersicht Teil 3 88
D.4. Marktübersicht Teil 4 89
D.5. Marktübersicht Teil 5 90
D.6. Marktübersicht Teil 6 91
D.7. Aussortierte Plattformen 92

E.1. Detaillierte Bewertung für das Ein-Personen-Projekt 100
E.2. Detaillierte Bewertung für das Software-Praktikum 101
E.3. Detaillierte Bewertung für das Studienprojekt 102
E.4. Detaillierte Bewertung für das interne Projekt 103
E.5. Bewertung JavaForge Bearbeiter 1 104
E.6. Bewertung GForge Bearbeiter 1 104
E.7. Bewertung JavaForge Bearbeiter 2 105
E.8. Bewertung GForge Bearbeiter 2 105

1. Einleitung

1.1. Hintergrund

Die Entwicklung vieler Software-Produkte, speziell im Open-Source-Bereich, findet heute geographisch und organisatorisch verteilt statt: die Beteiligten - Entwickler und Benutzer - sitzen in verschiedenen Winkeln der Erde, arbeiten zu unterschiedlichen Zeiten, haben unterschiedliche Verpflichtungen und Interessen. Zur Koordinierung der Entwicklung kommen aus diesem Grund Internet-basierte Kollaborationsplattformen zum Einsatz, welche die Zusammenarbeit und Kommunikation durch Werkzeuge unterstützen sollen. Zu diesen Werkzeugen gehören etwa Konfigurationsverwaltungssysteme, Bugtracker oder Diskussionsforen; zusätzlich können die Plattformen auch genutzt werden, um Informationen über die Software oder die Möglichkeit zum Download anzubieten.

Zu den häufig eingesetzten offenen Entwicklungsplattformen zählen unter anderem SourceForge.net, Launchpad, BerliOS und Google Code. Die Entscheidung für eine bestimmte Plattform hat Auswirkungen auf die weitere Entwicklung der Software: Die verfügbaren Werkzeuge bestimmen in vielen Fällen das Vorgehen. Bei der Auswahl einer geeigneten Plattform sollten Entwickler deshalb die von den verschiedenen Plattformen zur Verfügung gestellten Möglichkeiten vergleichen und die Eignung für den eigenen Entwicklungsprozess als Kriterium heranziehen.

1.2. Aufgabenstellung

Aufgabe der Fachstudie ist der Vergleich und die Bewertung verschiedener frei nutzbarer Kollaborationsplattformen unter Berücksichtigung unterschiedlicher Nutzungsszenarien und Anforderungsprofile. Ein Resultat der Fachstudie soll die Empfehlung einer Entwicklungsplattform für die Weiterentwicklung von liberAda sein. liberAda ist ein Dokumentationswerkzeug für Ada-Programme, das in der Abteilung SE entwickelt wird. Die Software wird von einem wissenschaftlichen Mitarbeiter der Abteilung entwickelt. Studenten können sich bei der Suche nach möglichen Fehlern beteiligen und Bugreports schreiben. Dabei geben wir nicht nur eine Empfehlung für das Projekt liberAda, sondern auch für weitere Nutzungsszenarien. Diese Nutzungsszenarien sind Studienprojekte, Softwarepraktika und interne

Projekte in der Industrie; durch eine Befragung verantwortlicher Entwickler (Maintainer) werden diese Nutzungsszenarien näher beleuchtet.

1.3. Leserkreis

Die Fachstudie richtet sich an die verantwortlichen Entwickler und Maintainer von Software-Projekten, die auf der Suche nach einer geeigneten Kollaborationsplattform für ihr Projekt sind. Die Fachstudie richtet sich außerdem an die Betreiber von Kollaborationsplattformen, da sie aus der Fachstudie Anforderungen gängiger Software-Projekte an Kollaborationsplattformen ableiten können.

1.4. Vorgehen

Zu Beginn der Fachstudie wurde ein Überblick über die verwandten Arbeiten erstellt. Dafür wurde eine Recherche zu verwandten Arbeiten durchgeführt. Die Ergebnisse der Recherche werden im Kapitel Verwandte Arbeiten aufgelistet.

Anschließend wurde die Fachstudie in drei Phasen aufgeteilt: Projekt, Kollaborationsplattform und Bewertung.

In der ersten Phase richtet sich der Fokus auf das Projekt. In einem Projekt wird Software entwickelt. Dabei werden die Entwickler durch die Werkzeuge einer Kollaborationsplattform, auf der das Projekt erstellt und betrieben wird, unterstützt. Eine ausführliche Definition wird im Kapitel Projektprofile gegeben. Da es sehr viele unterschiedliche Projekte gibt, werden die Projekte zunächst Profilen zugeordnet, die in der Fachstudie im Kapitel Projektprofile definiert werden. Die Zuordnung erfolgt anhand von Merkmalen der Projekte.

Im Hinblick auf eine Bewertung der Kollaborationsplattform ist es erforderlich, zu erfahren, welche Tätigkeiten in den einzelnen Projektprofilen durchgeführt werden. Dies wird durch ein Nutzungsszenario beschrieben. Hierfür wurden Projektrepräsentanten der einzelnen Projektprofile gesucht und mit ihnen ein Interview durchgeführt. Das Ergebnis ist ein Nutzungsszenario für ein konkretes Projekt, welches die Tätigkeiten angibt, die während des konkreten Projektes durchgeführt wurden. Im Kapitel Nutzungsszenarien werden die Nutzungsszenarien definiert und die Ergebnisse der Interviews aufgeführt.

Die zweite Phase der Fachstudie beschäftigt sich mit den Kollaborationsplattformen. Eine Kollaborationsplattform ist eine Internet-basierte Plattform mit Werkzeugen, die die Entwickler bei der Durchführung von Tätigkeiten während der Softwareentwicklung unterstützen. Durch eine Recherche im Internet wurden Kollaborationsplattformen für die Fachstudie ermittelt. In einer Marktübersicht werden die Plattformen mit ihren unterstützten Werkzeugen dargestellt. Da sich jedoch nicht alle Kollaborationsplattformen aus der Marktübersicht

für die Bewertung in der Fachstudie eignen, wurden Auswahlkriterien definiert. Diese Auswahlkriterien werden in der Fachstudie als K.O.-Kriterien bezeichnet. Nach Anwendung der K.O.-Kriterien auf die Marktübersicht, bleiben die Plattformen übrig, die für die Bewertung herangezogen werden; die Kurzliste. Anschließend werden die verbliebenen Plattformen anhand ihrer Werkzeuge miteinander verglichen. Im Kapitel Kollaborationsplattformen werden die K.O.-Kriterien definiert und die Kurzliste erstellt. Der Vergleich der Plattformen wird ebenfalls im Kapitel Kollaborationsplattformen durchgeführt.

Die dritte Phase führt die Ergebnisse der vorherigen Phasen zu einer Bewertung zusammen. Dafür wurde ein einheitliches Bewertungssystem, das Bewertungsschema, definiert, mit dem die Plattformen bewertet wurden. Das Bewertungsschema setzt sich aus den Nutzungsszenarien und dem Vergleich der Plattformen zusammen. Das Ergebnis der Bewertung ist eine Rangfolge, welche die Plattformen absteigend nach ihrer Funktionalität sortiert. Anschließend wird zusätzlich noch eine Bewertung der Usability und Integration der Werkzeuge der beiden bestplatzierten Plattformen durchgeführt, um eine qualitative Aussage über die Kollaborationsplattform und deren Werkzeuge machen zu können. Nach diesen Bewertungsschritten wird eine konkrete Empfehlung für die einzelnen Projektprofile gegeben. Die Definition des Bewertungsschemas, die Bewertung der Usability, Integration und die konkrete Empfehlung werden im Kapitel Bewertung der Plattformen durchgeführt.

Im Kapitel Fazit wird rückwirkend die Durchführung der Fachstudie zusammengefasst und ein Fazit daraus gezogen.

Die nächsten Abschnitte beschreiben das Vorgehen der Autoren während der Fachstudie. Sie geben dem Leser einen Überblick über die in dieser Fachstudie erarbeiteten Teilergebnisse. Die Teilergebnisse werden in den folgenden Kapiteln noch genauer betrachtet.

1.4.1. Verwandte Arbeiten

Zunächst wurde eine Literaturrecherche durchgeführt, um verwandte Arbeiten zu dem Thema „Kollaborative Entwicklungsplattformen“ zu finden. Diese Arbeiten dienen als Quellennachweis für Aussagen in der Fachstudie und für eine erste Informationsgrundlage bezüglich des Themas. Das Ergebnis der Recherche ist eine Sammlung von Arbeiten und deren Bewertung in Bezug auf die Relevanz für die Fragestellung der Fachstudie.

1.4.2. Projektmerkmale

Die Projektmerkmale wurden aus der Arbeit von Rainer und Gale: „Sampling open source projects from portals: some preliminary investigations“ [12] entnommen. Durch die Projektmerkmale wurden Eigenschaften definiert, die ein Projekt, in diesem Fall das Software-Projekt, besitzt. Werden alle Merkmale zusammengenommen, beschreiben sie in jeder Kon-

stellation ein konkretes Projektprofil. Die Projektmerkmale werden in der Studie benötigt, da aus ihnen Projektprofile generiert werden.

1.4.3. Tätigkeiten

Wie auch die Projektmerkmale kommen die Tätigkeiten aus Ergebnissen der Literaturrecherche, allerdings kommen sie aus mehreren Quellen wie z.B. von Josef Altmann und Gustav Pomberger: „Kooperative Softwareentwicklung: Konzepte, Modell und Werkzeuge“ [8]. Die Tätigkeiten beschreiben Aktivitäten, die in einem typischen Software-Projekt anfallen. Die in den Tätigkeiten aufgeführten Aktivitäten untergliedern sich in den Kategorien: Entwicklung, Qualitätssicherung, Projektverwaltung, Dokumentation und Kommunikation. Aus den Tätigkeiten wird dann in den folgenden Phasen der Interviewfragebogen erstellt. Sie fließen in das Nutzungsszenario mit ein und dienen als Vergleichskriterium für die einzelnen Plattformen.

1.4.4. Projektprofile

Die Projektprofile definieren eine Klasse von Projekten. Ein Projektprofil ist dabei eine mögliche Kombination der verschiedenen Ausprägungen der Projektmerkmale. Die Namensgebung der Projektprofile ist willkürlich und folgt keiner Konvention. Anhand der Projektklassen werden entsprechende repräsentative Projekte für das Interview gewählt. Außerdem richten sich die Nutzungsszenarien, die sich aus den Interviews ableiten, an die Projektprofile.

1.4.5. Nutzungsszenarien

Die im Interview befragten Personen arbeiten in Projekten, die sich als Repräsentanten für die einzelnen Projektprofile eignen. Das Interview zielt darauf ab, die Tätigkeiten zu erfassen, die in einem konkreten Projekt durchgeführt werden. Daraus wird ein allgemeines Nutzungsszenario für das entsprechende Projektprofil erstellt. Der im Interview verwendete Fragebogen beinhaltet die in einer früheren Phase erstellten Tätigkeiten. Das Ergebnis jedes Interviews ist ein Nutzungsszenario für die entsprechenden Projektprofile.

1.4.6. Marktübersicht

Eine Recherche im Internet hat sich angeboten, da die Kollaborationsplattformen Internetbasiert sind. Dazu wurde zunächst im Wikipedia Eintrag „Comparison of open source software hosting facilities“ [15] betrachtet. Sämtliche Plattformen, die dort aufgelistet sind,

werden in der Fachstudie verwendet. Die Dissertation von Haggen Hau Heng So: „Construction of an evaluation model for free/open source project hosting sites“ [3] bietet weitere Plattformen, die für die Fachstudie verwendet werden können.

Als nächstes wird durch eine Internetrecherche nach weiteren Plattformen gesucht. Es werden Schlüsselwörter für die Recherche verwendet, die mit dem Thema verwandt sind. Eine Auflistung der Schlüsselwörter befindet sich in Kapitel Verwandte Arbeiten.

Durch Verlinkungen auf Seiten der Kollaborationsplattformen sind neue Plattformen gefunden worden. Die Liste der Plattformen soll einen ersten Überblick darüber geben, wie viele Plattformen es im Internet bereits gibt. Sie bilden die Basis für die Marktübersicht. Die Liste erhebt keinen Anspruch auf Vollständigkeit, da im vorhandenen dynamischen Umfeld regelmäßig neue Kollaborationsplattformen in Betrieb gehen.

Die Marktübersicht leitet sich aus der Internetrecherche ab. Nach der Recherche, wird dem Leser ein Überblick über die Marktsituation gegeben. Dafür wird eine Tabelle erstellt, welche die gefundenen Plattformen auflistet. Die Zeilen beinhalten die Plattformen und die Spalten beinhalten die Eigenschaften, die eine Plattform besitzt. Diese Eigenschaften werden aus der Literatur [3, 13, 1] abgeleitet. Durch eine dreistufige Bewertung (vorhanden, teilweise vorhanden und nicht vorhanden), wird die Marktübersicht vervollständigt. Somit hat der Leser einen Überblick, welche Plattformen vorhanden sind und welche Eigenschaften sie in welcher Ausprägung besitzen.

1.4.7. K.O.-Kriterien

Die Marktübersicht gibt nur einen groben Überblick darüber, welche Plattformen vorhanden sind. Jedoch eigenen sich nicht alle Plattformen, um auf ihnen Projekte laufen zu lassen. Daher müssen die brauchbaren von den nicht brauchbaren Plattformen getrennt werden. Dies geschieht durch Anwendung von K.O.-Kriterien. Dafür wird zunächst festgelegt was eine Kollaborationsplattform alles können muss [12, 13, 3], damit auf ihr Projekte erstellt und betrieben werden können. Im Anschluss daran werden K.O.-Kriterien definiert. Diese K.O.-Kriterien werden auf die Marktübersicht angewendet. Dabei werden die Plattformen eliminiert, die die Kriterien nicht erfüllen. Das Resultat ist eine Kurzliste, die alle Plattformen enthält, die näher betrachtet werden.

1.4.8. Vergleichskriterien

Die Vergleichskriterien leiten sich aus den Tätigkeiten ab und dienen dem Vergleich der Plattformen. Die einzelnen Tätigkeiten fassen mehrere Plattformeigenschaften zusammen. Damit nun die Plattformen miteinander verglichen werden können, wie gut sie bestimmte Tätigkeiten unterstützen, werden Vergleichskriterien definiert. Dafür sind die einzelnen Werkzeuge zur Durchführung der Tätigkeiten auf den Plattformen der Kurzliste von jedem

Bearbeiter einzeln bewertet worden. Die Bewertung erfolgt in drei Stufen (wird voll unterstützt mit zusätzlichen Extras, wird unterstützt und wird überhaupt nicht unterstützt). Das Ergebnis ist eine Tabelle, welche die einzelnen Plattformen mit den Vergleichskriterien auflistet und die je nach Ausprägung bewertet werden. Die Vergleichskriterien bilden eine Grundlage für das Bewertungsschema.

1.4.9. Bewertungsschema

Das Bewertungsschema leitet sich aus den Nutzungsszenarien und den Vergleichskriterien ab. Das Bewertungsschema bildet die Grundlage, um die verglichenen Plattformen zu bewerten. Dafür werden die Tätigkeiten aus den einzelnen Nutzungsszenarien mit den Tätigkeiten in den Vergleichskriterien verglichen. Das Nutzungsszenario spiegelt das Anforderungsprofil eines Projektprofils wieder und beschreibt, welche Tätigkeiten enorm wichtig, unwichtig oder nur wichtig sind. Anhand dieser Wichtigkeit werden die einzelnen Plattformen herangezogen und bewertet, je nachdem wie gut sie eine Tätigkeit des Nutzungsszenarios unterstützen. Die Bewertung erfolgt in vier Abstufungen (+, o, - und - -). Das Ergebnis ist eine Tabelle mit der Bewertung der einzelnen Tätigkeiten auf der Plattform. Daraus lässt sich eine Rangfolge ableiten. Je mehr Tätigkeiten eine Plattform unterstützt, desto höher wird sie platziert. Die beiden besten Plattformen werden noch qualitativ bewertet, indem die Bearbeiter die Usability und Integration der Werkzeuge bewerten. Dafür werden Kriterien definiert. Die Bewertung erfolgt mit Punkten. Diese Bewertung ist subjektiv. Sie soll dem Leser aber zumindest einen Eindruck von der Qualität der verwendeten Werkzeuge vermitteln. Das Bewertungsschema ist Grundlage für die konkrete Empfehlung.

1.4.10. Empfehlung

Die Empfehlung leitet sich aus dem Bewertungsschema und der Kurzliste ab. Anhand der Bewertung der Plattformen aus der Kurzliste ist eine Rangfolge entstanden, welche die Plattformen absteigend nach der Anzahl der unterstützenden Tätigkeiten auflistet. Diese Auflistung wird für alle Nutzungsszenarien erstellt, sodass am Ende eine konkrete Plattform als Empfehlung für ein Nutzungsszenario dient.

2. Verwandte Arbeiten

In diesem Kapitel werden die Ergebnisse der Literaturrecherche behandelt. Dafür wird zunächst recherchiert, ob es verwandte Arbeiten zu dem Thema „Kollaborative Entwicklungsplattformen“ gibt. Dazu wird im Internet und in der Fachliteratur recherchiert. Die verwandten Arbeiten werden als Grundlage für die weitere Bearbeitung der Fachstudie herangezogen und im Text referenziert.

2.1. Vorgehensweise

Die Arbeit an der Fachstudie beginnt mit einer Literaturrecherche. Das Ziel ist es, verwandte Arbeiten zu dem Thema „Kollaborative Entwicklungsplattformen“zu finden. Die verwandten Arbeiten geben eine erste Anleitung, wie in der Fachstudie vorgegangen werden sollte und dienen als Informationsgrundlage zu diesem Thema. Damit die Bearbeiter unabhängig voneinander die Recherche durchführen können, wird zunächst definiert, nach welchen Schlüsselwörtern und wo gesucht wird.

Es wird in der Fachliteratur recherchiert, indem eine Suche in der Bibliothek durchgeführt wird.

Für die Recherche im Internet werden folgende drei für die Suche nach Referenzliteratur verwendet:

- GoogleScholar[1]
- ACM Portal[2]
- IEEE Xplore[3]

[1] http://scholar.google.com
[2] http://portal.acm.org
[3] http://ieeexplore.ieee.org

2.2. Schlüsselwörter

Für die Einschränkung der Suche und um möglichst viele themenverwandte Ergebnisse zu erhalten, werden die folgenden Schlüsselwörter für die Internetrecherche verwendet:

software opensource hosting, sourceforge, googlecode, sourceforge googlecode, project hosting, opensource development, web-based software development, software development, hosting opensource, community opensource, community software, development groupware, plattform sourceforge googlecode, plattform googlecode, opensource development web 2.0, distributed software development, software project hosting, collaborative development environment

Anhand dieser Schlüsselwörter wird in den oben aufgelisteten Internetseiten nach Referenzliteratur gesucht. Die Liste der Schlüsselwörter bezieht sich auf die Suche während der Recherchephase. Die erhaltenen Ergebnisse werden im nächsten Abschnitt aufgelistet und kurz erläutert, welche konkreten Informationen sie enthalten und wozu sie in der Fachstudie verwendet werden.

2.3. Suchergebnisse

Vor der Recherche ist der Eindruck entstanden, dass es zum Thema „Kollaborative Software-Entwicklungsplattformen“ sehr wenige Publikationen und Arbeiten gibt. Die Recherche hat diesen ersten Eindruck bestätigt. Es war schwierig, verwandte und brauchbare Arbeiten zu finden, sodass nach der Recherche nur wenige Ergebnisse gefunden wurden, die weiter verwendet werden. Das ist ein weiteres Indiz dafür, dass bisher wenig auf diesem Gebiet bearbeitet wurde und die Fachstudie neue Erkenntnisse auf diesem Gebiet bringt.

Im Folgenden werden die verwertbaren Ergebnisse aufgelistet und kurz erläutert, welche Informationen sie für die Fachstudie enthalten.

Dieter Spath, Sven Schimpf, Andreas Kugler (2007): Webbasierte Open-Source-Kollaborationsplattformen [13]

Diese Studie der Fraunhofer Gesellschaft beschäftigt sich mit der Open-Source Entwicklung in Teams und die Nutzung von Kollaborationsplattformen in solchen Entwicklungen. Dafür werden Charakteristika eines Open-Source-Projektes definiert und Vor- und Nachteile aufgelistet. Damit ein Open-Source-Projekt auf einer Kollaborationsplattform laufen kann, müssen bestimmte Voraussetzungen erfüllt sein. Diese Voraussetzungen werden genannt. Als Resultat der Studie wird ein Vergleich mehrerer Plattformen durchgeführt und diese anhand von vorher festgelegten Kriterien bewertet.

Diese Studie kommt der Fachstudie sehr nahe, jedoch werden in der Studie von Dieter Spath et al. keine Projektprofile betrachtet. Die Studie bildet für die Fachstudie den Leitfaden, um

einen Ansatz für die Vorgehensweise zu erhalten. Viele Bereiche, wie die Bewertung oder die Kriterien für die Bewertung werden in der Fachstudie referenziert.

Austen Rainer, Stephen Gate (2005): Sampling open source projects from portals: some preliminary investigations [12]

Das Dokument befasst sich mit Projekten, die auf SourceForge gehostet werden. Das Dokument wertet die Daten von 50000 Projekten unter verschiedenen Kriterien aus. Dafür wird das populärste Projekt auf SourceForge mit einer Datensammlung aller Projekte, die sich aus den Ergebnissen eines Crawlers zusammensetzt, verglichen, um so eine Art Nutzungsprofil von SourceForge zu erstellen.

Die Definition einer Plattform und deren Vorteile für eine Projekterstellung sind für die Fachstudie relevant und werden als Referenz herangezogen. Für die Fachstudie spielt die Auswertung der einzelnen Projekte keine Rolle.

Haggen Hau Heng So (2005): Construction of an Evaluation Model for Free/Open Source Project Hosting Sites [3]

Die Doktorarbeit befasst sich mit der Erstellung eines Bewertungsmodells für kollaborative Plattformen. Dafür wird beschrieben, wie ein Open-Source-Projekt aufgebaut ist, um daraus Anforderungen an die Kollaborationsplattformen zu stellen. Diese Kollaborationsplattformen werden dann bewertet, um so Aussagen treffen zu können, was eine Kollaborationsplattform alles können muss, damit man auf ihre Open-Source-Projekte erstellen und betreiben kann.

Diese Publikation wird für die Erstellung des Bewertungsschemas verwendet. Zusätzlich werden einige Plattformen in der Arbeit miteinander verglichen. Aus diesem Vergleich werden die Plattformen, sowie Vergleichskriterien extrahiert.

Grady Booch, Alan W. Brown (2003): Collaborative Development Environment [5]

Dieses Buch beschreibt die Entwicklung der kollaborativen Entwicklungsumgebungen, stellt verschiedene Typen der Plattformen und listet Eigenschaften auf, die kollaborative Plattformen erfüllen sollten, damit Projekte auf ihr laufen können. Dafür werden folgende drei Kategorien betrachtet: Koordination, Kollaboration und Soziale Plattform. Diese drei Kategorien fassen unterschiedliche Eigenschaften einer Plattform zusammen.

Diese Information wird dafür verwendet, um die Anforderungen an die Plattformen zu erstellen, die wir für unsere engere Auswahl benötigen und um eine Liste der Eigenschaften zu haben, die für Kollaborationsplattformen wichtig sind.

Karl Fogel (2005): Producing Open Source Software: How to Run a Successful Free Software Project [1]

Dieses Buch beschreibt das Vorgehen, um ein erfolgreiches Open-Source-Projekt zu starten und durchzuführen. Dabei werden Werkzeuge aufgelistet, die für die Durchführung notwendig sind. Im Dokument wird in einem seperaten Abschnitt auf die Doktorarbeit von Haggen

Hau Heng So [3] verwiesen und macht einen Vorschlag, welche kollaborative Plattform man für das Hosting verwenden soll.

Die genannten Werkzeuge werden für die Marktübersicht verwendet, um einen Überblick zu geben, welche Plattform welche Werkzeuge zur Verfügung stellt.

Josef Altmann, Gustav Pomberger (1999): Kooperative Softwareentwicklung: Konzepte, Modell und Werkzeuge [8]

Diese Publikation beschreibt Probleme bei einer kooperativen Softwareentwicklung. Es werden Konzepte, Modelle und Werkzeuge vorgestellt, wie mehrere Projektmitarbeiter intensiv zusammenarbeiten können, obwohl sie zeitlich und räumlich verteilt sind. Dabei wird auf die einzelnen Tätigkeiten im Softwareentwicklungsprozess eingegangen und anhand von Werkzeugen erläutert, wie eine kooperative Entwicklung möglich ist.

Die einzelnen Tätigkeiten werden in der Fachstudie für das Interview verwendet, damit daraus ein Nutzungsszenario abgeleitet werden kann. Damit wird gezeigt in welcher Art von Projekt, welche Tätigkeiten durchgeführt werden.

Laurie Ann Williams (2000): Collaborative Software Process [16]

Diese Dissertation beschäftigt sich mit dem kollaborativen Softwareentwicklungsprozess und zeigt die Unterschiede zwischen einem einzelnen Entwickler und einer kollaborativen Entwicklung mit mehreren Entwicklern.

Hier wird auch auf einzelne Tätigkeiten im Softwareentwicklungsprozess eingegangen, die für das Interview in der Fachstudie verwendet werden, damit daraus ein Nutzungsszenario abgeleitet werden kann.

2.3.1. Erkenntnisse aus den Suchergebnissen

Ein Resultat der Recherche ist, dass es auf diesem Gebiet wenige Arbeiten gibt und niemand bisher einen Vergleich von 48 Plattformen gemacht hat. Die Fraunhofer-Studie [13] beschäftigt sich lediglich mit 17 Plattformen, die auf einem eigenen Server aufgesetzt werden müssen. Die Dissertation von Haggen Hau Heng So: „Construction of an Evaluation Model for Free/Open Source Project Hosting Sites“ [3] beschäftigt sich nur mit zehn Kollaboratinsplattformen. Der Artikel von Filippo Lanubile, Christof Ebert, Rafael Prikladnicki und Aurora Vizcaino: „Collaboration Tools for Global Software Engineering“ [6] vergleicht nur neun Plattformen anhand bestimmter Kriterien. Arbeiten, die eine ähnlich hohe Zahl verschiedener Kollaborationsplattformen betrachtet haben, konnten bei der Literaturrecherche nicht gefunden werden.

3. Projektprofile

In diesem Kapitel wird der Begriff Projekt definiert, der in der Fachstudie verwendet wird. Anschließend werden einige Merkmale, die ein Software-Projekt besitzt, aufgezeigt und schließlich anhand dieser Merkmale Projektprofile definiert. Das Ziel der Fachstudie ist die Abgabe einer konkreten Empfehlung für das Projekt liberAda. Projekte sind verschieden. Eine allgemeingültige Empfehlung kann nicht gegeben werden. Deshalb werden zunächst ähnliche Projekte zusammengefasst. Die Zusammenfassung der Projekte spiegelt sich in den definierten Projektprofilen wider. Für jedes Projektprofil wird eine konkrete Empfehlung abgegeben.

3.1. Definition Projekt

In der Literatur gibt es zahlreiche Definitionen eines Projektes. Ludewig und Lichter beschreiben im Buch „Software Engineering“ [9] ein Projekt dadurch, dass es zeitlich begrenzt ist, einen Erzeuger und einen Zweck hat und der Abnehmer der Kunde ist. Diese Definition ist für Open-Source Projekte schwer anzuwenden, daher wird der Begriff Projekt in der Fachstudie wie folgt definiert:

Ein Projekt besteht aus einer Reihe von Aktivitäten, die zur Entwicklung einer Software ausgeführt werden. Das Projekt wird auf einer Kollaborationsplattform im Internet angelegt und betrieben. Jedes Projekt hat einen Projekteigentümer, der das Projekt auf einer Plattform erstellt hat. Es können mehrere Entwickler an einem Projekt beteiligt sein. Das Projekt ist üblicherweise zeitlich nicht beschränkt.

3.2. Projektmerkmale von Software-Projekten

Projekte unterscheiden sich je nach Aufgabe und Bedingungen in Ihren Merkmalen. Austen Rainer und Stephen Gale [12] haben die folgenden Merkmale festgelegt:

- Entwicklungsstatus
- Umgebung
- Zielgruppe
- Anzahl der Entwickler

- Lizenzierung
- Betriebssystem
- Programmiersprache
- Thema
- Sprache
- Gewinnorientierung
- Räumliche Verteilung

Es werden die Projektmerkmale ausgewählt, die relevant für die Auswahl einer Kollaborationsplattform sind. Diese werden für die Ableitung der Projektprofile im weiteren Verlauf der Fachstudie verwendet. Projektprofile fassen mehrere Projekte mit gleichen Projektmerkmalen zusammen. In der folgenden Tabelle sind für die Projektprofile relevante Projektmerkmale [12] aufgeführt. Dabei werden lediglich die Merkmale berücksichtigt, die für die Auswahl einer Entwicklungsplattform ausschlaggebend sind.

Name	**Beschreibung**	**Ausprägungen**
Anzahl der Entwickler	Ein Entwickler ist eine Person, die Software entwickelt. Je nach Rolle sind folgende Tätigkeiten auszuführen: Spezifizieren, Entwerfen, Testen, Prüfen und Verwalten [9].	Eine Person (1) [E], Klein (2-3) [K], Mittel (4-12) [M], Mittelgroß(13-20) [MG], Groß (20 +) [G]
Räumliche Verteilung	Räumliche Verteilung der Projektbeteiligten. Zwei Ausprägungen sind dabei relevant: Die Entwickler können sich zu einer vereinbarten Zeit und an einem vereinbarten Ort treffen oder nicht.	Räumlich verteilt oder nicht räumlich verteilt
Lizenzierung	Die Zugänglichkeit des Programmcodes. Ist der Programmcode öffentlich und somit für jeden lesbar oder wird der Programmcode unter Verschluss gehalten, sodass nur die internen Entwickler Zugriff auf den Code haben?	Open-Source oder Closed-Source
Gewinnorientierung	Wird die Software gewinnorientiert entwickelt oder wird sie frei zur Verfügung gestellt.	Gewinnorientiert (kommerziell) und nicht gewinnorientiert (nicht kommerziell)

Tabelle 3.1.: Projektmerkmale

3.3. Projektprofile nach Merkmalen

Die Klassifikation der einzelnen Projekte leitet sich aus den vorher definierten Projektmerkmalen ab. In der folgenden Tabelle werden die Projektprofile nach den Ausprägungen der verschiedenen Projektmerkmale gebildet. Durch diese Klassifikation werden ähnliche und gleiche Projekte zusammengefasst. Die Auflistung beruht nicht auf Vollständigkeit und demonstriert lediglich die Variationsmöglichkeiten. Die Profile, die in der Fachstudie näher betrachtet werden, werden in der Tabelle grau hervorgehoben. Für diese Projektprofile wird in den nachfolgenden Kapiteln eine konkrete Empfehlung abgegeben. Die Namensgebung ist willkürlich und folgt keiner Konvention, da es keine allgemein anerkannte Taxonomie der Projektarten gibt.

Projektmerkmale / Projektprofile	Entwickler	Verteilung	Lizenzierung
Ein-Personen-Projekt	E	N	Closed Source
Software-Praktikum	K	N	Closed Source
Studienprojekt	M	J	Closed Source
Großes Projekt	MG	J	Open Source
Internationales Projekt	G	J	Closed Source
Internes Projekt	MG	N	Closed Source

Tabelle 3.2.: Projektprofile nach Merkmalen

Ein-Personen-Projekt: Im Ein-Personen-Projekt gibt es nur einen Entwickler. Die Entwicklung findet nicht verteilt statt und die Projekte sind Closed Source.

Softwarepraktikum: Softwarepraktika, die an der Universität Stuttgart durchgeführt werden, werden in einer Gruppe von drei Entwicklern durchgeführt. Die Entwicklung findet nicht verteilt statt und die Software ist Closed Source.

Studienprojekt: Studienprojekte, die an der Universität Stuttgart durchgeführt werden, werden in einer Gruppe von sechs bis zwölf Entwicklern durchgeführt. Die Entwicklung findet verteilt statt, da manche Entwickler nur von zu Hause aus arbeiten. Auch hier ist die Software Closed Source.

Großes Projekt: Bei einem großen Projekt sind 13-20 Entwickler beteiligt und die Entwicklung findet verteilt statt, da die Entwickler geographisch verteilt sind. Die Projekte sind Open Source und so für andere Entwickler offen.

Internationales Projekt Bei einem internationalen Projekt sind mehr als 20 Entwickler beteiligt. Die Entwicklung findet geographisch verteilt statt. Die Software ist dabei Closed Source.

Internes Projekt: Das interne Projekt hat 13-20 Entwickler und findet oft intern in einem Unternehmen statt, sodass es keine geographische Verteilung gibt. Das Projekt ist ebenfalls Closed Source.

Im weiteren Verlauf der Fachstudie werden für die einzelnen Projektprofile Projektrepräsentanten gesucht, um mit ihnen ein Interview durchzuführen. Dieses Interview dient dazu herauszufinden, welche Tätigkeiten in diesem konkreten Projekt durchgeführt werden. Die beiden Projektprofile Großes Projekt und Internationales Projekt werden im weiteren Verlauf der Fachstudie nicht weiter betrachtet, da sich keiner der angeschriebenen Repräsentanten gemeldet hat. Die angeschriebenen Repräsentanten sind in Tabelle 3.3 verzeichnet.

Projekt	URL
Miktex	http://miktex.org
TexnicCenter	http://www.texniccenter.org
eMule	http://emule.org
Notepad++	http://notepad-plus-plus.org
PidGin	http://www.pidgin.im

Tabelle 3.3.: Repräsentanten für Interviews

4. Nutzungsszenarien

In diesem Kapitel werden Nutzungsszenarien erstellt, die das Vorgehen innerhalb eines Projektes beschreiben. Dafür werden Tätigkeiten definiert, die während des Softwareentwicklungsprozesses durchgeführt werden. Um festzustellen, welche Tätigkeiten in den zuvor definierten Projekten ausgeführt werden, werden Interviews mit Projektrepräsentanten der entsprechenden Projektprofile durchgeführt. Das Ergebnis dieser Interviews ist die Gewichtung der einzelnen Tätigkeiten für ein bestimmtes Projektprofil, das Nutzungsszenario. Die Nutzungsszenarien bilden eine Grundlage für die Bewertung der Plattformen.

4.1. Tätigkeiten

Im Softwareentwicklungsprozess gibt es Tätigkeiten, die auszuführen sind, um Software zu entwickeln oder zu verändern [9]. Außerdem bedarf es einem hohen Maß an Kooperation, da die Entwicklung häufig zeitlich und räumlich verteilt stattfindet [8]. Die Ausführung von Tätigkeiten unterscheidet sich je nach Anforderung des Projekts. Je nach Projektprofil ist eine Tätigkeit wichtiger als eine andere. Zum Beispiel spielt die Wartung im Softwarepraktikum eine nicht so wichtige Rolle, wie bei einem internen Projekt. Diese unterschiedlichen Einstufungen der Wichtigkeit einer Tätigkeit für ein Projektprofil werden im Nutzungsszenario festgehalten.

Diese Tätigkeiten werden in der Tabelle 4.1 definiert und kurz erläutert. In der Abbildung 4.1 werden die Tätigkeiten veranschaulicht, die in einem Softwareentwicklungsprozess durchgeführt werden. Für die Fachstudie werden die Tätigkeiten in fünf Kategorien zusammengefasst: Entwicklung, Projektverwaltung, Kommunikation, Dokumentation und Qualitätssicherung.

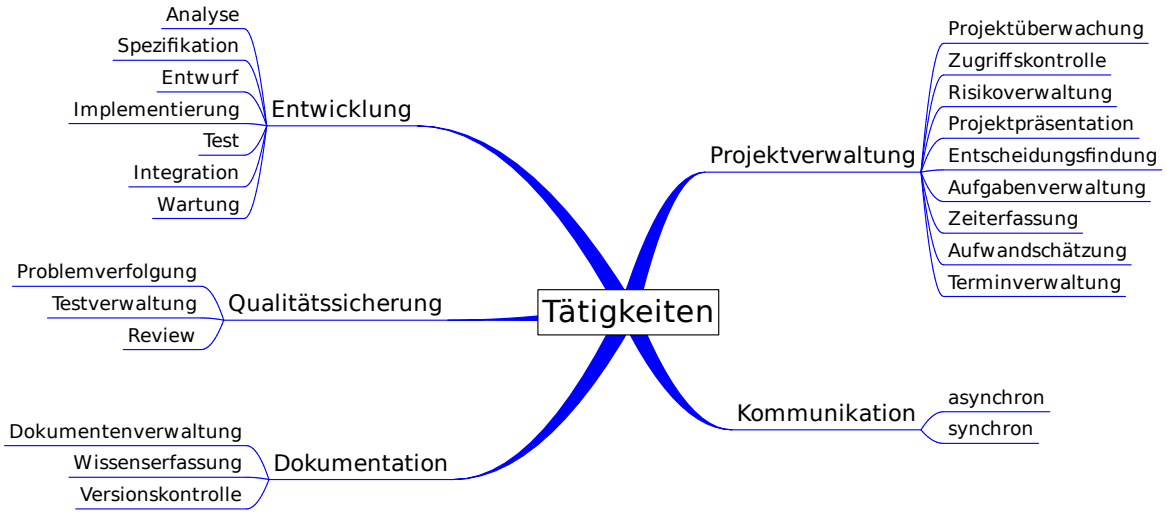


Abbildung 4.1.: Taxonomie der Tätigkeiten

Entwicklung	
Tätigkeit	**Beschreibung**
Analyse	In der Analyse wird das bestehende Problem durchdrungen, um es zu verstehen [9]. Es wird eine Ist- und eine Soll-Analyse durchgeführt. Es werden die Anforderungen an die zu erstellende Software dokumentiert und Analysetechniken wie Befragung, Beobachtung, Interview oder Prototyping angewendet.
Spezifikation	In der Spezifikation werden die in der Analyse festgestellten Anforderungen geordnet, dokumentiert, geprüft, ergänzt und korrigiert.
Entwurf	Im Entwurf werden Schnittstellen der zu entwickelnden Komponenten definiert. Das Gesamtsystem, welches die Kollaboration der Komponenten definiert, wird beschrieben.
Implementierung	In der Implementierung werden die im Entwurf festgelegten Komponenten in der gewählten Programmiersprache codiert.
Test	Im Test werden die Randbedingungen definiert, die Eingaben systematisch ausgewählt und die Ergebnisse dokumentiert und nach Kriterien beurteilt, die vor dem Test festgelegt wurden.

Integration	In der Integration wird aus Codefragmenten das Programm generiert. Einzelne Komponenten werden zu einer Gesamtstruktur vereint.
Wartung	In der Wartung werden entdeckte Fehler des laufenden Systems behoben und neue Anforderungen umgesetzt.
Projektverwaltung	
Tätigkeit	**Beschreibung**
Projektüberwachung	In der Projektüberwachung werden projektspezifische Informationen und Daten gesammelt, um sie in einer bestimmten Visualisierungsform darzustellen.
Zugriffskontrolle	In der Zugriffskontrolle werden Zugriffsregeln auf Projektressourcen für beteiligte Personen definiert und verwaltet.
Risikoverwaltung	In der Risikoverwaltung werden Risiken identifiziert, analysiert, bewertet und verfolgt. Des weiteren werden Gegenmaßnahmen geplant.
Projektpräsentation	In der Projektpräsentation wird der Fortschritt des Projekts und die Projektinhalte veröffentlicht.
Entscheidungsfindung	In der Entscheidungsfindung wird über relevante Themen abgestimmt, um eine demokratische Einigung zu erhalten.
Aufgabenverwaltung	In der Aufgabenverwaltung werden die einzelnen Arbeitspakete erfasst und den Projektbeteiligten zugeordnet.
Zeiterfassung	In der Zeiterfassung wird der Arbeitsaufwand der einzelnen Projektbeteiligten protokolliert und zentral festgehalten.
Aufwandsschätzung	In der Aufwandsschätzung werden alle Aufgaben erfasst und der Aufwand für deren Bearbeitung abgeschätzt.
Terminverwaltung	In der Terminverwaltung werden projektrelevante Termine zentral erfasst und verwaltet.
Kommunikation	
Tätigkeit	**Beschreibung**
Asynchrone Kommunikation	In der asynchronen Kommunikation werden ortsunabhängig und zeitunabhängig Nachrichten ausgetauscht.
Synchrone Kommunikation	In der synchronen Kommunikation werden ortsunabhängig und zeitabhängig Nachrichten ausgetauscht.

Dokumentation	
Tätigkeit	**Beschreibung**
Dokumentenverwaltung	In der Dokumentenverwaltung werden Dokumente, die während dem Entwicklungsprozess entstehen, verwaltet. Die Dokumente werden in elektronischer Form erstellt und einer Konfigurationsverwaltung unterstellt.
Wissenserfassung	In der Wissenserfassung wird Fachwissen, das entsteht, festgehalten und zentral verwaltet. Es werden Informationen kategorisiert und Taxonomien gebildet.
Versionskontrolle	In der Versionskontrolle werden alle Versionen und Revisionen der Dokumente, wie dem Code, der Spezifikation, dem Entwurf, der Tests usw. festgehalten und verwaltet.
Qualitätssicherung	
Tätigkeit	**Beschreibung**
Problemverfolgung	In der Problemverfolgung werden Fehler analysiert, registriert, geprüft und behoben.
Testverwaltung	In der Testverwaltung werden die angelegten Testfälle abgelegt und bearbeitet.
Review	Im Review werden Dokumente, die zur Prüfung ausstehen, von Gutachtern bewertet und positive Befunde dokumentiert.

Tabelle 4.1.: Tätigkeiten im Softwareentwicklungsprozess

4.1.1. Werkzeugtabelle

Die folgende Tabelle listet für die oben erwähnten Tätigkeiten Werkzeuge auf, die der Tätigkeit eindeutig zugeordnet werden können. Diese Werkzeugtabelle dient dazu, die einzelnen Eigenschaften und Funktionen der Plattformen zu erfassen und aufzulisten. Konkrete Ausprägungen der Werkzeuge werden in `Maschinenschrift` hervorgehoben. Tätigkeiten, denen kein eindeutiges Werkzeug zugeordnet werden kann oder für die es keine Werkzeugunterstützung gibt, werden in *kursiver Schrift* beschrieben.

Tätigkeit	**Werkzeug**
Analyse	Texteditor
Spezifikation	Texteditor, UML-Editor
Entwurf	Texteditor, UML-Editor
Implementierung	Entwicklungsumgebung (IDE)

Integration	Buildsystem
Wartung	*Ein Werkzeug zur Erfassung, zentralen Ablage und Korrektur der gefundene Fehler sowie zur systematischen Anpassung der Software an neue und geänderte Anforderungen*
Projektüberwachung	Dashboard
Zugriffskontrolle	Access Control List (ACL)
Risikoverwaltung	*Ein Werkzeug zur Risikoerfassung bzw. Risikobewertung*
Projektpräsentation	*Ein Werkzeug zur textuellen und visuellen Präsentation des Softwareprodukts*
Entscheidungsfindung	Umfragetool
Aufgabenverwaltung	Ticketsystem, Task-Tracker, Time-Tracker
Aufwandschätzung	Time-Tracker,
Terminverwaltung	Kalender
Asynchrone Kommunikation	Forum, News, Nachrichtenfunktion, E-Mail
Synchrone Kommunikation	Instant Messaging, `IRC, Jabber`
Dokumentenverwaltung	*Ein Werkzeug zur zentralen Verwaltung (Erstellen, Bearbeiten, Löschen) von Dokumente*
Wissenserfassung	Wiki
Versionskontrolle	Versionsverwaltungssystem, `Subversion`
Problemverfolgung	Bugtracker
Testverwaltung	*Ein Werkzeug zur zentralen Verwaltung und Archivierung der Testfälle*
Review	*Ein Werkzeug zur Erfassung der Befunde am Prüfling, sowie der Gewichtung der Befunde*

Tabelle 4.2.: Werkzeugtabelle

Die definierten Tätigkeiten werden dazu verwendet, um den Kern des Interviews zu gestalten. Im folgenden Abschnitt wird auf die Interview weiter eingegangen.

4.2. Interviews

In diesem Abschnitt werden die Interviews aufgelistet, die während der Fachstudie durchgeführt werden.

Der Zweck des Interviews besteht darin, herauszufinden, welche Tätigkeiten in konkreten Projekten durchgeführt werden. Dazu werden Projektrepräsentanten der einzelnen Projektprofile gesucht, um mit ihnen ein Interview durchzuführen. Aus den Antworten der Befragten werden Nutzungsszenarien abgeleitet, die genau beschreiben, welche Tätigkeiten im konkreten Projekt durchgeführt werden.

4.2.1. Interviewdurchführung

Aufbau des Fragenkatalogs

Der Fragenkatalog wird mit Hilfe des Fragebogen-Leitfadens [2] erstellt. Die Fragen sind in drei Abschnitte gegliedert. Der erste Abschnitt besteht aus Fragen bezüglich des Projektes. Hier werden allgemeine Fragen zum Projekt gestellt. Der nächste Abschnitt besteht aus Fragen bezüglich der Tätigkeiten, die während des Projektes durchgeführt werden. Der letzte Abschnitt umfasst Fragen bezüglich sonstiger Tätigkeiten, die eventuell nicht mit dem ersten Abschnitt abgedeckt werden. Die Fragen sind als offene Fragen formuliert. Der vollständige Fragenkatalog befindet sich im Kapitel Interview-Fragenkatalog

Arten der Interviewdurchführung

Die Interviewpartner werden persönlich befragt. Der Fragebogen besteht aus vielen offenen Fragen, die darauf abzielen, detaillierte Informationen über den Entwicklungsprozess im Projekt herauszuarbeiten. Details zum Projekt können am besten bei persönlichen Gesprächen ermittelt werden.

Persönlich Es wird ein Termin für ein persönliches Interview mit dem Projektrepräsentanten ausgemacht. Das Interview wird maximal 45 Minuten dauern. In diesem Interview sind zwei bis drei Bearbeiter anwesend, um Missverständnisse beim Protokollieren der Antworten zu vermeiden. Dem Interviewpartner werden die Fragen aus dem Fragebogen gestellt. Die Antworten werden schriftlich festgehalten.

Durchführung

Für die Interviews mussten zunächst Interviewpartner gefunden werden, die über ihre Erfahrungen im Umgang in ihren Projekten berichten konnten. Insgesamt zehn mögliche Interviewpartner wurden angeschrieben; davon erklärten sich vier bereit, an einem Interview teilzunehmen.

Nachdem in vorherigen Arbeitspaketen die Projektprofile erstellt wurden, wurden die Plattformen anhand bestimmter Kriterien ausgewählt. Auf den verbliebenen Plattformen wurden repräsentative Projekte gemäß unserer Projektprofile ausgewählt. Dabei wurden nur aktive und populäre Projekte ausgewählt. Dadurch waren Alternativen gegeben, falls ein Projekt nicht auf die Einladung zum Interview reagiert hat.

In einem nächsten Schritt wurden die Interviewpartner angeschrieben, um die Studie und die beteiligten Mitarbeiter vorzustellen, den Zweck des Interviews zu erläutern und um einen Termin zu vereinbaren, an dem das Interview durchgeführt wurde. Ein persönliches Interview ist zu bevorzugen und sollte nach Möglichkeit rechtzeitig mit dem Interviewpartner vereinbart werden. Um dem Interviewpartner Zeit für eine angemessene Vorbereitung auf das Interview zu geben, wurde ihm der Fragebogen vorab per E-Mail zugesendet.

Für das Ein-Personen-Projekt wurde das Projekt liberAda, welches in der Abteilung SE der Universität Stuttgart von Ivan Bogicevic entwickelt wird, auf Grund der Aufgabenstellung, gewählt. LiberAda ist ein internes Projekt zur Generierung von Ada Kommentaren. Diese Kommentare werden ähnlich wie in JavaDoc durch spezielle Konstrukte gekennzeichnet und dann generiert. Die generierten Kommentare werden dann als HTML exportiert.

Für das Software-Praktikum (SoPra) wurde ein Student des Softwaretechnik Studiengangs der Universität Stuttgart als Interviewpartner, welcher am Projekt Horos teilgenommen hat, gewählt. Horos ist ein Review Assistent, der die Gutachter und Moderatoren eines Reviews bei der Erstellung des Fragebogens behilflich ist. Die erstellten Fragebögen können importiert und exportiert werden und auf anhand der Fragen werden die Befunde gegenüber dem Prüfling festgestellt.

Für das Studienprojekt (StuPro) wurde ein Student des Softwaretechnik Studiengangs der Universität Stuttgart als Interviewpartner, welche am Projekt ConfiSS teilgenommen hat, gewählt. ConfISS ist eine Open Source Webanwendung zur Verwaltung von Konferenzen. Mit dem System können wissenschaftliche Tagungen (Konferenzen), inklusive der Einreichung und Begutachtung der Einreichungen, einfach und komfortabel verwaltet werden.

Für das interne Projekt wurde das Projekt Bauhaus gewählt. Das Interview wurde mit dem Entwickler Thomas Eisenbarth durchgeführt. Bauhaus ist ein Forschungsprojekt der Universität Stuttgart mit dem Ziel eine Lösung zum erkennen von Software-Erosion, bei Projekten mit langer Entwicklungsdauer, zu entwickeln. Die Software sucht nach totem Code, Copy and Paste Fehlern, verstößen gegen den Entwurf und weiteren Fehlern die sich in einem langen Entwicklungsprozess einschleichen.

Diese Interviewpartner wurden gewählt, da sie an Projekten arbeiten, die den einzelnen Projektklassen entsprechen.

Die Ergebnisse werden im nächsten Abschnitt aufgelistet.

4.2.2. Interviewergebnisse

Die Interviewergebnisse werden mit Hilfe eines Interviewprotokolls festgehalten. Die Interviewergebnisse geben Auskunft darüber, welche Tätigkeiten in den einzelnen konkreten Projekten durchgeführt wurden. Die konkreten Ergebnisse und die einzelnen Protokolle befinden sich im Anhang C: Interview-Ergebnisse.

4.3. Nutzungsszenarien

Die Interview-Ergebnisse werden nun dazu verwendet, um festzulegen, wie wichtig die einzelnen Tätigkeiten für das Projekt sind. Dabei werden drei Stufen unterschieden und wie folgt gekennzeichnet:

Sehr wichtig	wichtig	unwichtig
●	◐	○

Tabelle 4.3.: Einstufung der Tätigkeiten

Dabei bedeutet unwichtig, dass eine Tätigkeit nicht durchgeführt wird. Der Unterschied zwischen wichtig und sehr wichtig liegt darin, dass eine Tätigkeit die als sehr wichtig eingestuft wird, auf jeden Fall im Projekt durchgeführt werden muss. Die wichtige Tätigkeit muss nicht zwangsweise durchgeführt werden.

Durch diese Einstufung leitet sich ein Nutzungsszenario für jede einzelne Projektklasse ab. Nachfolgend werden zusammenfassend alle Nutzungsszenarien aufgelistet. Die Nutzungsszenarien werden für die spätere Bewertung der Plattformen bezüglich der Projektklasse herangezogen.

	liberAda	Horos	ConfiSS	Bauhaus
Analyse	◐	●	●	●
Spezifikation	◐	●	●	●
Entwurf	◐	●	●	●
Implementierung	●	●	●	●
Test	●	●	●	●
Integration	◐	◐	●	●
Wartung	◐	○	◐	●
Problemverfolgung	◐	◐	◐	●
Testverwaltung	○	◐	◐	◐
Review	○	●	●	◐
Dokumentenverwaltung	○	◐	◐	◐
Wissenserfassung	○	○	●	◐
Versionskontrolle	◐	●	●	●
Asynchron	○	◐	◐	●
Synchron	○	●	●	●
Projektüberwachung	◐	◐	◐	●
Zugriffskontrolle	◐	◐	●	◐
Risikoverwaltung	○	○	○	◐
Projektpräsentation	◐	◐	◐	●
Entscheidungsfindung	○	◐	◐	◐
Aufgabenverwaltung	○	◐	●	●
Zeiterfassung	◐	●	●	◐
Aufwandsabschätzung	◐	◐	◐	◐
Terminverwaltung	○	◐	●	●

Tabelle 4.4.: Ausgeführte Tätigkeiten in den Projektprofilen (Nutzungsszenario)

5. Kollaborationsplattformen

In diesem Kapitel wird eine Marktübersicht der bestehenden Plattformen gegeben, dazu wird zunächst der Begriff Kollaborationsplattform als Basis für die weitere Verwendung definiert. Anschließend werden K.O.-Kriterien zur Auswahl geeigneter Plattformen definiert, da nicht alle Plattformen der Markübersicht für die weitere Betrachtung in Frage kommen. Das Ergebnis der Anwendung der K.O.-Kriterien auf die Markübersicht ist eine Liste von Plattformen, die für die weitere Betrachtung und Empfehlung verwendet werden.

5.1. Definition

Eine Kollaborationsplattform ist eine Internet-basierte Webseite, auf der Projekte angelegt werden können. Dadurch wird eine zentrale Anlaufstelle für die Entwickler geschaffen,um sie beim kollaborativen Entwicklungsprozess zu unterstützen. Für den Entwicklungsprozess werden Werkzeuge auf der Kollaborationsplattform zur Verfügung gestellt. Wichtig ist dabei, dass die Kollaborationsplattformen über das Internet erreichbar sein müssen, ohne vorher eine Anwendung installieren zu müssen.

In der Fachstudie wird diese Definition verwendet. Dabei können Open-Source und Closed-Source Softwareprojekte auf der Kollaborationsplattform angelegt werden.

5.2. Marktübersicht

Grundlage der Marktübersicht ist der englische Wikipedia-Eintrag „Comparison of open source software hosting facilities“ [15]. In diesem Eintrag wird bereits ein Vergleich verschiedener Kollaborationsplattformen durchgeführt.

Im Wiki-Eintrag wird auf die Dissertation von Haggen Hau Heng So [3]) verwiesen, die Plattformen miteinander vergleicht. Zum Teil überschneiden sich die Plattformen, mit denen aus dem Wikipedia Eintrag.

Die Marktübersicht besteht zum großen Teil aus den Informationen aus dem Wikipedia-Eintrag und aus der verlinkten Webseite. Weitere Plattformen, die weder im Wikipedia Eintrag, noch in der Dissertation aufgelistet werden, sind aus mehreren Google-Suchen

hervorgegangen. Dabei bezog sich die Google-Suche hauptsächlich auf die Schlüsselwörter, die im Kapitel Verwandte Arbeiten verzeichnet sind. Die Marktübersicht besteht aus 48 Kollaborationsblattformen, sie befindet sich detailliert im Anhang D, Marktübersicht. Des weiteren wurden 46 Eigenschaften und Merkmale gefunden, die eine Plattform besitzen kann. In der Tabelle D.1 beschreiben die Zeilen die Plattformen und die Spalten beschreiben deren Eigenschaften und Merkmale.

Im folgenden Abschnitt werden die K.O.-Kriterien aufgelistet, die auf die Plattformen angewendet werden

5.3. K.O.-Kriterien

Die K.O.-Kriterien dienen dazu, solche Plattformen aus der Liste zu entfernen, die gewisse Minimalanforderungen nicht erfüllen und deshalb für den ernsthaften Einsatz nicht in Betracht kommen. Daraus resultiert eine minimierte Plattformliste, deren Plattformen näher betrachtet und im weiteren Verlauf der Fachstudie bewertet werden.

5.3.1. Anforderungen

Eine Plattform muss bestimmte Anforderungen erfüllen, damit man auf ihr kollaborative Projekte durchführen kann. Booch und Brown definieren die drei Anforderungen Koordination, Kollaboration und Community [5]. Diese drei Anforderungen fassen mehrere Eigenschaften zusammen. So bezieht sich die Anforderung Koordination auf die Konfigurationsverwaltung, Dashboard, Kalender usw. Foren, virtuelle Konferenzräume und weitere Eigenschaften werden unter Kollabortion zusammengefasst. Unter Community werden die persönlichen Einstellmöglichkeiten für die einzelnen Entwickler zusammengefasst.

Diese Anforderungen von Booch und Brown stellen eine Mindest- bzw. Maximalanforderung an Plattformen, die sie erfüllen müssen, damit sie weiter betrachtet werden.

Für die Fachstudie sind nicht alle Anforderungen relevant. Zunächst werden die Anforderungen aufgestellt, die für die Fachstudie wichtig sind. Aus diesen Anforderungen leiten sich die K.O.-Kriterien ab. Diese werden im weiteren Verlauf der Fachstudie auf die Plattformen angewendet, um die Plattformen auszusortieren, die nicht den K.O.-Kriterien genügen.

In Abbildung 5.1 werden die Anforderungen and die Kollaborationsplattformen veranschaulicht.

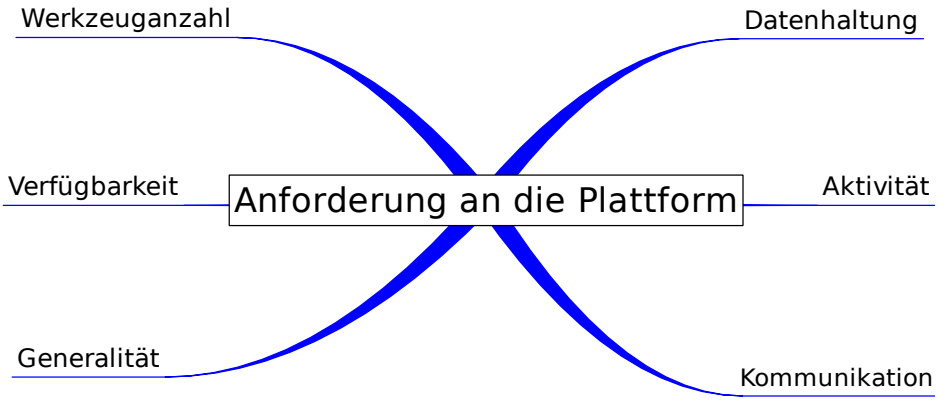


Abbildung 5.1.: Anforderung an die Plattform

Datenhaltung Eine Plattform muss mehreren Entwicklern, die am Softwareentwicklungsprozess beteiligt sind, die Möglichkeit bieten, Dokumente (insbesondere der Code), die erstellt werden, zu verwalten. Das bedeutet, dass auf der Plattform Daten zentral abgespeichert und abgerufen werden. Dabei muss das Abrufen dezentral möglich sein, da die Entwickler eventuell in Projekten geographisch verteilt arbeiten. Die Daten müssen vor unbefugten Zugriffen geschützt werden. Eine Zugriffsverwaltung regelt die Rechte innerhalb einer Plattform. Diese Anforderung wird unter dem Begriff Datenhaltung zusammengefasst.

Kommunikation Eine Plattform muss den beteiligten Entwicklern die Möglichkeit bieten, synchron und asynchron miteinander zu kommunizieren, damit sie untereinander zeitunabhängig und ortsunabhängig Informationen austauschen können. Je höher die Zahl der Entwickler, desto komplexer wird der Kommunikationsweg. Diese Anforderung wird unter dem Begriff Kommunikation zusammengefasst.

Generalität Eine Plattform muss, unabhängig von der Anwendung und der Programmiersprache, die Möglichkeit bieten, jede Art von Projekt zu hosten. Diese Unbahängigkeit ist für einen Vergleich der Plattformen, der auf Basis von allgemeingültigen Aussagen gemacht wird, notwendig. Zustätzlich muss die Plattform mindestens die Sprache Englisch oder Deutsch unterstützen, da die englische Sprache im Software-Engineering weit verbreitet ist und die Fachstudie auf deutsch geschrieben ist. Diese Anforderung wird unter dem Begriff Generalität zusammengefasst.

Verfügbarkeit Eine Plattform muss zu jeder Zeit über das Internet erreichbar sein, damit Entwickler orts- und zeitunabhängig auf ihre Projekte zugreifen können. Die Plattform sollte durch einen Provider zur Verfügung gestellt werden, damit Entwickler ad-hoc Projekte

erstellen und verwalten können, ohne die Plattform selbst auf einem Server zu installieren. Die Plattform kann jedoch optional eine Standalone-Version zur Verfügung stellen. Diese Anforderung wird unter dem Begriff Verfügbarkeit zusammengefasst.

Werkzeuganzahl Eine Plattform muss mindestens vier Werkzeuge bieten, damit sie als Kollaborationsplattform und nicht als ein spezielles Werkzeug betrachtet wird. Der Begriff Werkzeug wird im Kapitel Nutzungsszenarien, festgelegt. Diese Anforderung wird unter dem Begriff Werkzeuganzahl zusammengefasst.

Die definierten K.O.-Kriterien werden im folgenden Abschnitt auf die Marktübersicht angewendet, um die Plattformen auszusortieren, die nicht den K.O.-Kriterien genügen. Die weiter zu betrachtenden Plattformen werden in einer Kurzliste aufgenommen.

5.4. Kurzliste

In der Tabelle 5.1 werden die Plattformen aufgelistet, die nach Anwendung der K.O.-Kriterien in die engere Auswahl fallen. Diese Plattformen werden in den weiteren Kapiteln näher betrachtet.

Name der Plattform	Url
Assembla	https://www.assembla.com/
BerliOS	http://www.berlios.de/
Bitbucket	http://bitbucket.org/
CodePlex	http://www.codeplex.com/
CodeSpaces	http://www.codespaces.com/
GForge	http://gforge.org/gf/
Gitorious	http://gitorious.org/
Gna.org	http://Gna.org
GNU Savannah	http://savannah.gnu.org/
GoogleCode	http://code.google.com/
GoPlan	http://goplanapp.com/
Huddle	http://www.huddle.net/
JavaForge	http://www.javaforge.com
Launchpad	https://launchpad.net/
Origo	http://www.origo.ethz.ch
Ourproject.org	http://ourproject.org

Sarovar	http://sarovar.org
ShareSource	http://sharesource.org/
SourceForge	http://sourceforge.net/
Unfuddle	http://unfuddle.com/

Tabelle 5.1.: Kurzliste

Die Plattformen, die nicht den K.O.-Kriterien genügen, sind im Anhang D: K.O-Kriterien angefügt. Es wird beschrieben, um welche Plattform es sich handelt und welches Kriterium nicht erfüllt wurde. Insgesamt sind so 26 Plattformen durch die K.O.-Kriterien aussortiert worden.

Im nachfolgenden Abschnitt wird ein Vergleich der Plattformen in der Kurzliste durchgeführt.

5.5. Vergleich

Der Zweck dieses Abschnitts ist es objektive Kriterien und Merkmale für Kollaborationsplattformen auszuarbeiten, um diese anschließend miteinander vergleichen zu können. Dazu werden Vergleichskriterien für Kollaborationsplattformen aufgestellt und ihre Merkmale, sowie Ausprägungen beschrieben.

Die im Kapitel Nutzungsszenarien erstellten Tätigkeiten bilden die Basis für die Vergleichskriterien. Die Tätigkeiten fassen somit einzelne Plattformeigenschaften zusammen und stellen die Vergleichskriterien dar.

Um einen einheitlichen Vergleich zu gewährleisten, werden die einzelnen Tätigkeiten auf den Plattformen aus der Kurzliste durchgeführt. Dies geschieht anhand einem strikten Vorgehensmodell. Das Vorgehensmodell beschreibt dabei ein Szenario, welches auf allen Plattformen durchgeführt wird. Dafür wird auf der Plattform jede Tätigkeit durchgeführt. Um eine Bewertung abgeben zu können, wird in der Tabelle 5.3 definiert, wann eine Tätigkeit rudimentär oder voll unterstützt wird. Dabei werden die Tätigkeiten bewertet. Das vollständige Vorgehensmodell befindet sich im Anhang D: Vorgehensmodell, des Anhangs. Falls eine Tätigkeit von einer Plattform auf keine Art und Weise unterstützt wird, wird sie mit einem ○ gekennzeichnet. Falls eine Tätigkeit rudimentär unterstützt wird, wird sie mit einem ◐ gekennzeichnet, falls eine Tätigkeit voll unterstützt wird, wird sie mit einem ● gekennzeichnet. In der Tabelle 5.2 wird die Bewertung im Vergleich anschaulich dargestellt.

○	keine Unterstützung
◐	rudimentäre Unterstützung
●	volle Unterstützung

Tabelle 5.2.: Bewertung im Vergleich

Analyse	◐	Ein Editor zur Auflistung der Anforderungen
	●	Ein Texteditor zur Auflistung und zur Priorisierung der Anforderungen
Spezifikation	◐	Ein Editor zur Dokumentation der Spezifikation
	●	Ein Texteditor zur Dokumentation der Spezifikation mit einer zusätzlichen graphischen Notation
Entwurf	◐	Ein Editor zur Dokumentation des Entwurfs
	●	Ein Texteditor zur Dokumentation des Entwurfs mit einer zusätzlichen graphischen Notation
Implementierung	◐	Ein Editor zur Dokumentation des Codes und zum Ausführen des Codes
	●	Ein Editor zur Dokumentation des Codes mit Texthighlighting. Der dokumentierte Code kann ausgeführt und nach Fehlern untersucht werden
Test	◐	Ein Editor zum Erstellen von Testfällen
	●	Ein Assistent zur Durchführung der Testfälle oder eine automatische Durchführung der Testfälle
Integration	◐	Ein Editor zum Erstellen von Skripten, die manuell angestoßen werden
	●	Ein Buildsystem zum automatischen Bauen und Integrieren des Systems
Wartung	Da die Tätigkeit Wartung als ein Sammelbegriff für den gesamten Entwicklungsprozess anzusehen ist, wird sind nicht weiter betrachtet	
Problemverfolgung	◐	Ein Editor zur textuellen Erfassung festgestellter Fehler und eine Übersicht der Fehler
	●	Ein Bugtracker mit Statusanzeige und Integration in andere Werkzeuge (Forum, SVN etc.)
Testverwaltung	◐	Eine Übersicht der erstellten Testfälle
	●	Eine Übersicht der erstellten Testfälle mit der zusätzlichen Möglichkeit, Testfälle zu bearbeiten und zu erstellen
Review	◐	Ein Editor zur Erfassung der Befunde

	●	Einen Assistenten für die Zuweisung der Prüflinge an die Gutachter mit einer Verwaltung der Befunde
Dokumentenverwaltung	◐	Eine Übersicht der erstellten Dokumente
	●	Eine Übersicht der Dokumente mit der zusätzlichen Möglichkeit Dokumente zu bearbeiten und zu erstellen
Wissenserfassung	◐	Ein Editor zur textuellen Wissenserfassung
	●	Ein Editor zur textuellen Wissenserfassung mit zusätzlichen Verlinkungen plattformweit
Versionskontrolle	◐	Eine manuelle Erfassung der einzelnen Versionen
	●	Eine automatische Erfassung der einzelnen Versionen mit zusätzlicher Versionshistorie
Kommunikation	◐	Es wird entweder die synchrone (Instant Messaging) oder die asynchrone Kommunkation (Email, Forum) unterstützt
	●	Es wird die synchrone (Instant Messaging) und die asynchrone Kommunkation (Email, Forum) unterstützt
Projektüberwachung	◐	Eine textuelle Übersicht über Projektdaten (aktive Download, Anzahl der Bugs)
	●	Eine visuelle Übersicht (Diagramme) über Projektdaten
Zugriffskontrolle	◐	Eine Passwortabfrage zur Sicherung der privaten Projektdaten
	●	Eine Passwortabfrage zur Sicherung der privaten Projektdaten und der Definition von projektspezifischen Rollen
Risikoverwaltung	◐	Ein Editor zur Auflistung der Risiken
	●	Ein Editor zur Auflistung der Risiken und eine Übersicht über die eingetretenen Risiken
Projektpräsentation	◐	Eine Möglichkeit Projektseiten zu erstellen mit projektspezifischem Inhalt
	●	Eine Möglichkeit Projektseiten aus Vorlagen zu erstellen und mit projektspezifischem Inhalt zu verknüpfen
Entscheidungsfindung	◐	Ein Editor zur Erfassung der Entscheidung
	●	Ein Editor zur Erfassung der Entscheidung und zur Abgabe der eigenen Entscheidung
Aufgabenverwaltung	◐	Ein Editor zur Erfassung der Aufgaben
	●	Ein Editor zur Erfassung der Aufgaben und der Zuweisung der Aufgaben an Mitarbeiter
Zeiterfassung	◐	Ein Editor zur Erfassung der aufgewendeten Zeit
	●	Ein Editor zur Erfassung der aufgewendeten Zeit und der Kategorisierung der Aufwände
Aufwandsschätzung	◐	Ein Editor zur Erfassung der Aufwände

	●	Ein Editor zur Erfassung der Aufwände und eine graphische Darstellung des Aufwands
Terminverwaltung	◐	Ein Kalender zur Organisation der Termine
	●	Ein Kalender zur Organisation der Termine und der Synchronisation mit anderen Entwicklern

Tabelle 5.3.: Rudimentäre und volle Unterstützung der Tätigkeiten

Die Tabelle 5.4 veranschaulicht den Vergleich der einzelnen Plattformen aus der Kurzliste.

Kollaborationsplattform	Analyse	Spezifikation	Entwurf	Implementierung	Test	Integration	Problemverfolgung	Testverwaltung	Review	Dokumentenverwaltung	Wissenserfassung	Versionskontrolle	Asynchron	Synchron	Projektüberwachung	Zugriffskontrolle	Risikoverwaltung	Projektpräsentation	Entscheidungsfindung	Aufgabenverwaltung	Zeiterfassung	Aufwandsschätzung	Terminverwaltung
Assembla	○	○	○	○	○	●	◐	○	○	◐	●	●	●	●	●	◐	○	●	○	●	●	○	◐
BerliOS	●	◐	◐	○	○	○	●	◐	○	●	●	●	●	○	●	●	○	◐	●	●	○	○	○
Bitbucket	○	○	○	○	○	○	◐	○	○	◐	●	●	◐	○	○	●	○	○	○	○	○	○	○
CodePlex	○	○	○	○	○	○	◐	◐	○	○	●	●	●	○	◐	●	○	◐	◐	●	○	○	○
CodeSpaces	○	○	○	○	○	○	●	○	○	◐	●	●	●	○	●	◐	○	○	○	●	●	●	○
GForge	○	○	○	○	○	●	●	○	○	◐	●	●	●	○	●	●	○	●	○	●	●	◐	○
Gitorious	○	○	○	○	○	○	●	○	◐	◐	●	●	○	○	◐	●	○	◐	○	○	○	○	○
Gna.org	○	○	○	○	○	○	●	○	○	●	◐	●	◐	○	●	●	○	◐	●	●	○	○	○
GNU Savan.	○	○	○	○	○	○	●	○	○	●	◐	●	◐	○	●	●	○	◐	●	●	○	○	○
GoogleCode	○	○	○	○	○	○	●	○	◐	◐	●	●	○	○	◐	●	○	◐	○	○	○	○	○
GoPlan	○	○	○	○	○	○	○	○	○	◐	○	○	●	○	◐	◐	○	○	○	●	●	○	●
Huddle	●	◐	◐	○	○	○	◐	◐	◐	●	●	◐	●	○	◐	●	○	◐	◐	●	○	◐	●
JavaForge	●	●	●	○	○	●	●	○	○	◐	●	●	●	●	●	●	○	◐	○	●	●	◐	○
Launchpad	●	◐	◐	○	○	○	●	◐	◐	●	●	●	●	○	●	◐	○	◐	◐	◐	○	◐	○
Origo	●	◐	◐	○	○	○	●	◐	○	●	●	●	●	○	●	◐	○	◐	○	●	○	○	○
Ourproject	○	○	○	○	○	○	●	◐	○	●	●	●	●	○	◐	●	○	◐	●	●	○	○	○
Sarovar	○	○	○	○	○	○	●	◐	○	●	○	●	●	○	◐	●	○	◐	●	●	○	○	○
SourceForge	○	○	○	○	○	○	●	◐	○	○	●	●	●	○	●	◐	○	◐	○	◐	○	○	○
Unfuddle	●	◐	◐	○	○	○	◐	◐	○	●	●	●	●	○	◐	●	○	◐	◐	●	○	○	◐

Tabelle 5.4.: Vergleich der Plattformen

Die Tabelle 5.4 zeigt, dass einige Tätigikeiten von vielen Plattformen unterstützt werden, jedoch werden andere Tätigkeiten von keiner Plattform unterstützt. Zu diesen nicht unterstützten Tätigkeiten gehören die Implementierung, der Test, und die Risikoverwaltung. Die Implementierung wird von keiner Plattform unterstützt, da diese Tätigkeit mit Hilfe einer Entwicklungsumgebung durchgeführt wird. Auch der Test wird häufig in der Entwicklungsumgebung, in der die Implementierung stattfindet, durchgeführt, weshalb auch hier keine zwingende Unterstützung von der Plattform notwendig ist. Jedoch bieten einige Plattformen eine Testverwaltung an, mit deren Hilfe man Testfälle bearbeiten und löschen kann. Bei der Risikoabschätzung spielt die Erfahrung der Entwickler eine große Rolle, sodass es schwierig ist, mit Hilfe eines Werkzeugs die Risiken abzuschätzen. Hinzu kommt das bei vielen Projekten keine Risikoabschätzung stattfindet. Dieses sind mögliche Gründe, warum die Plattformen kein Werkzeug zur Risikoverwaltung anbieten.

Weiterhin ist auffällig, dass die frühen Phasen des Softwareentwicklungsporzesses, Analyse, Spezifikation und Entwurf, nur von wenigen Plattformen unterstützt wird. Von den 22 Plattformen aus der Kurzliste, bieten nur sechs Plattformen eine Unterstützung für diese drei Phasen. Das kann daran liegen, dass für diese Phasen ein vollwertiger Texteditor mit graphischen Elementen, zum Zeichnen von Diagrammen, benötigt wird. Eine Möglichkeit, dass solch ein Editor nur von wenigen Plattformen zur Verfügung gestellt wird, könnte der Grund sein, dass hierfür spezielle, eigenständige Werkzeuge existieren, z.B. OpenOffice[1], VisualParadigm[2]. JavaForge ist die einzige Plattform, die alle drei Tätigkeiten voll unterstützt. Mit JavaForge ist es sogar möglich, graphische Elemente, wie Klassendiagramme, Use-Case-Diagramme usw., zu erstellen.

Die Tätigkeit Integration, wird lediglich von drei Plattformen unterstützt. Eine mögliche Begründung hierfür kann der hohe Speicherbedarf, sowie die hohe Prozessorbelastung für den Server der Plattform sein. Die Prozessorbelastung und der Speicherbedarf würden auch noch mit der Anzahl der Projekte steigen.

Dass es für das Review nur sehr wenige Werkzeuge gibt, spiegelt sich auch auf den Plattformen wider, da sehr wenige Plattformen keine Werkzeugunterstützung für das Review anbieten.

Das die synchrone Kommunikation (Chat) nur auf zwei Plattformen angeboten wird, kann daran liegen, dass die Entwickler meist nicht zur selben Zeit arbeiten und somit der Bedarf einer synchronen Kommunikation nicht hoch ist.

Bei der Tätigkeit Terminverwaltung haben wir keine Erklärung, weshalb diese Tätigkeit von so wenigen Plattformen (drei Plattformen) angeboten wird.

[1] http://de.openoffice.org
[2] http://www.visual-paradigm.com

Die Tätigkeit Problemverfolgung, Dokumentenverwaltung, Wissenserfassung, Versionskontrolle, asynchrone Kommunikation, Projektüberwachung, Zugriffskontrolle, Projketpräsentation und Aufgabenverwaltung werden nahezu von allen Plattformen unterstützt. Diese Tätigkeiten lassen sich im Gegensatz zu den frühen Phasen sehr gut verteilen. Zum Beispiel können zwei Aufgaben unabhänig voneinander erstellt werden, die sich auf ein Teilproblem beziehen, wohingegen bei der Spezifikation ein Überblick über das gesamte Dokument erforderlich ist, um die Spezifikation zu erstellen. So ist es z.B. notwendig die Anforderungen zu kennen, um das Mengengerüst zu erstellen.

Im nachfolgendem Evaluationsprozess wird anhand der Vergleichskriterien ein Bewertungsschema definiert, das zur Bewertung der einzelnen Plattformen herangezogen wird.

6. Bewertung der Plattformen

In diesem Kapitel wird die Eignung der vorher verglichenen Plattformen für verschiedene Nutzungsszenarien bewertet. Die Bewertung wird mit Hilfe der Nutzungsszenarien durchgeführt. Dafür werden die Tätigkeiten der Nutzungsszenarien mit den Tätigkeiten, die auf der Plattformen durchgeführt werden können, verglichen. Für die Bewertung wird eine dreistufige Gewichtung vorgenommen, die je nach Ausprägung unterschiedlich angewendet wird. Durch die Bewertung entsteht eine Rangfolge, welche die Plattformen absteigend nach der Anzahl der erfüllten Tätigkeiten sortiert. Um eine qualitative Aussage der angebotenen Werkzeuge auf der Plattform zu geben, werden die beiden bestplatzierten Plattformen qualitativ bewertet. Dazu werden die Integration und Usability der Werkzeuge von zwei Autoren der Fachstudie unabhängig voneinander bewertet und je nach Ausprägung unterschiedlich gewichtet. Anschließend wird für jedes Projektprofil eine konkrete Empfehlung abgegeben.

6.1. Bewertung

Während des Vergleichs der einzelnen Plattformen trat das Problem auf, dass es einige Tätigkeiten gab, die entweder teilweise oder nicht von den Plattformen unterstützt wurden. Daher werden diese Tätigkeiten aus der Bewertung ausgeschlossen, da es nicht sinnvoll ist, Tätigkeiten die von keiner Plattform unterstützt werden, zu bewerten. Dies betrifft vor allem folgende Tätigkeiten des Softwareentwicklungsprozesses:

- Analyse
- Spezifikation
- Entwurf
- Implementierung
- Test
- Integration
- Testverwaltung
- Review
- Synchrone Kommunikation

- Risikoverwaltung
- Terminverwaltung

Folgende Tätigkeiten werden bei der Bewertung berücksichtigt:

- Problemverfolgung
- Dokumentenverwaltung
- Wissenserfassung
- Asynchrone Kommunikation
- Projektüberwachung
- Zugriffskontrolle
- Projektpräsentation
- Entscheidungsfindung
- Aufgabenverwaltung
- Zeiterfassung
- Aufwandsschätzung

6.1.1. Bewertungsschema

Im folgenden Abschnitt wird das Bewertungsschema definiert, welches die Grundlage der Bewertung ist.

Die verbliebenen Tätigkeiten werden für den Vergleich herangezogen, indem die zuvor definierten Nutzungsszenarien mit den Tätigkeiten verglichen werden. Dafür wird eine Skala angelegt, die den Grad der Erfüllung der einzelnen Tätigkeiten zu den Nutzungsszenarien beschreibt. Dabei beschreibt - - den Abstand, dass eine Tätigkeit im Nutzungsszenario als sehr wichtig (●) eingestuft wird, aber von der Plattform nicht unterstützt wird. Wird eine Tätigkeit im Nutzungsszenario als sehr wichtig eingestuft (●) und die Plattform unterstützt diese Tätigkeit nur teilweise (◐), wird die Tätigkeit mit - bewertet. Dies gilt auch, wenn die Tätigkeit im Nutzungsszenario als wichtig (◐) eingestuft wird und die Plattform diese Tätigkeit nicht unterstützt. Dann wird die Tätigkeit ebenfalls mit einem - bewertet. Falls die Plattformtätigkeit und die die Tätigkeit im Nutzungsszenario übereinstimmen, wird die Tätigkeit mit einem = bewertet. Wird eine Tätigkeit im Nutzungsszenario als wichtig (◐) eingestuft und unterstützt die Plattform diese Tätigkeit und bietet noch weitere Zusatzoptionen, wird sie mit einem + bewertet. Das gilt auch dafür, wenn eine Tätigkeit im Nutzungsszenario als unwichtig (○) eingestuft wird und die Plattform die Tätigkeit nur teilweise bzw. voll unterstützt. Die Tabelle 6.1 veranschaulicht das Bewertungsschema.

	Nutzungszenario		
Plattform	●	◐	○
●	=	+	+
◐	-	=	+
○	- -	-	=

Tabelle 6.1.: Bewertungsschema

Anschließend werden die einzelnen Bewertungen gezählt, sodass die Gesamtzahl der einzelnen Bewertungsklassen für jede Plattform sichtbar ist. Damit nun eine Aussage getroffen werden kann, welche Plattform am besten von der Funktionalität zu einem Nutzungsszenario passt, werden die Bewertungsklassen + und = zusammengezählt, da dies bedeutet, dass diese Tätigkeiten von der Plattform erfüllt bzw. übererfüllt werden. Die Plattform mit den meisten Punkten in diesen beiden Bewertungsklassen passt von der Funktionalität her am besten zum Nutzungsszenario und wird auf Platz eins der Plattformrangfolge gesetzt. Die restlichen Plattformen werden dann absteigend nach diesen Punkten sortiert. Haben zwei oder mehrere Plattformen die gleiche Anzahl an + und = wird zunächst die Anzahl der - - betrachtet. Die Plattform deren Anzahl an - - höher ist, wird in der Rangfolge niedriger positioniert. Sollte die Anzahl der - - ebenfalls gleich sein, werden die Plattform in der Rangfolge gleich positioniert.

Die oben beschriebene Bewertung soll anhand des folgenden Beispiels in Tabelle 6.2 veranschaulicht werden. Dazu wird die Plattform Assembla und das Nutzungszenario des Ein-Personen-Projekts gewählt.

Tätigkeiten	Assembla	Ein-Personen-Projekt	Bewertung
Problemverfolgung	◐	◐	=
Dokumentenverwaltung	◐	○	+
Wissenserfassung	●	○	+
Versionskontrolle	●	◐	+
Asynchron	●	○	+
Projektüberwachung	●	◐	+
Zugriffskontrolle	◐	◐	=
Projektpräsentation	●	◐	+
Entscheidungsfindung	○	○	=
Aufgabenverwaltung	●	○	+
Zeiterfassung	●	◐	+
Aufwandsschätzung	○	◐	-

Tabelle 6.2.: Beispiel der Bewertung

Die folgenden Abschnitte listen die einzelnen Rangfolgen der jeweiligen Nutzungsszenarien auf. Dafür werden die Vergleichskriterien und die Nutzungsszenarien herangezogen und mit Hilfe des Bewertungsschemas wird die Rangfolge ermittelt.

6.1.2. Ein-Personen-Projekt

In der Tabelle 6.3 wird die Rangfolge der Plattformen für das Ein-Personen-Projekt aufgelistet. Die vollständige Bewertung befindet sich im Kapitel Bewertung des Anhangs.

Platz	Plattform	+	=	-	- -
1	GForge	7	5	1	0
1	JavaForge	7	5	1	0
2	Assembla	6	5	2	0
2	CodeSpaces	9	2	2	0
3	BerliOS	9	2	1	1
3	GNA.org	8	3	1	1
3	Gnu Savannah	8	3	1	1
3	Huddle	6	5	1	1
3	ourproject.org	8	3	1	1
3	Sarovar	7	4	1	1
3	Unfuddle	6	5	1	1
4	Launchpad	6	4	2	1
4	Origo	7	3	2	1
5	GoPlan	2	7	3	1
5	SourceForge	5	4	3	1
6	Gitorious	4	4	4	1
6	GoogleCode	4	4	4	1
7	Bitbucket	3	4	5	1
7	CodePlex	5	2	5	1

Tabelle 6.3.: Rangfolge für das Ein-Personen-Projekt

6.1.3. Software-Praktikum

In der Tabelle 6.4 wird die Rangfolge der Plattformen für das Software-Praktikum aufgelistet. Die vollständige Bewertung befindet sich im Kapitel Bewertung des Anhangs.

Platz	Plattform	+	=	-	- -
1	GForge	10	3	0	0
1	JavaForge	9	4	0	0

2	Assembla	8	4	1	0
2	CodeSpaces	9	3	1	0
2	Huddle	6	6	1	0
2	Launchpad	8	4	1	0
3	BerliOS	8	3	2	0
3	CodePlex	6	5	2	0
3	Gitorious	5	6	2	0
3	GNA.org	9	2	2	0
3	Gnu Savannah	9	2	2	0
3	GoogleCode	5	6	2	0
3	Origo	7	4	2	0
3	ourproject.org	8	3	2	0
3	Sarovar	7	4	2	0
3	SourceForge	6	5	2	0
3	Unfuddle	7	4	2	0
4	Bitbucket	5	4	4	0
4	GoPlan	4	5	4	0

Tabelle 6.4.: Rangfolge für das Software-Projekt

6.1.4. Studienprojekt

In der Tabelle 6.5 wird die Rangfolge der Plattformen für das Studienprojekt aufgelistet. Die vollständige Bewertung befindet sich im Kapitel Bewertung des Anhangs.

Platz	Plattform	+	=	-	- -
1	GForge	4	8	1	0
1	JavaForge	3	9	1	0
2	BerliOS	5	6	1	1
2	Huddle	2	9	1	1
2	ourproject.org	4	7	1	1
2	Unfuddle	2	9	1	1
3	Assembla	5	5	3	0
3	CodeSpaces	4	6	3	0
4	CodePlex	1	9	2	0
4	GNA.org	4	6	2	1
4	Gnu Savannah	4	6	2	1

4	Launchpad	4	6	2	1
5	Sarovar	3	7	1	2
6	Origo	4	5	3	1
7	Gitorious	1	7	3	2
7	GoogleCode	1	7	3	2
8	Bitbucket	2	5	4	2
8	SourceForge	3	4	4	2
9	GoPlan	1	5	5	2

Tabelle 6.5.: Rangfolge für das Studienprojekt

6.1.5. Internes-Projekt

In der Tabelle 6.6 wird die Rangfolge der Plattformen für das interne Projekt aufgelistet. Die vollständige Bewertung befindet sich im Kapitel Bewertung des Anhangs.

Platz	Plattform	+	=	-	- -
1	GForge	4	7	2	0
2	JavaForge	3	7	3	0
3	CodeSpaces	3	7	2	1
4	BerliOS	4	5	4	0
4	Launchpad	2	7	4	0
5	Assembla	1	7	5	0
5	GNA.org	3	5	5	0
5	Gnu Savannah	3	5	5	0
5	Origo	2	6	5	0
5	ourproject.org	4	4	5	0
6	Huddle	3	4	6	0
6	Sarovar	3	4	6	0
6	Unfuddle	3	4	6	0
7	CodePlex	2	4	7	0
7	SourceForge	1	5	7	0
8	Gitorious	2	3	6	2
8	GoogleCode	2	3	6	2
9	GoPlan	1	4	5	3
10	Bitbucket	2	2	6	3

Tabelle 6.6.: Rangfolge für das Interne-Projekt

Die Bewertung der einzelnen Plattformen bezüglich der Nutzungsszenarien hat ergeben, dass GForge und JavaForge immer auf den ersten beiden Plätzen sind. In der Summe der übererfüllten und erfüllten Tätigkeiten haben die beiden Plattformen immer dieselbe Anzahl. Nur beim internen Projekt erfüllt GForge eine Tätigkeit mehr als JavaForge. Diese Tätigkeit ist die Entscheidungsfindung. Somit werden für die qualitative Bewertung GForge und JavaForge herangezogen. Ebenfalls ist zu erwähnen, dass GoPlan und Bitbucket häufig die letzten beiden Plätze belegen. In der Mitte der Rangfolge belegen die restlichen Plattformen, je nach Nutzungsszenario, unterschiedliche Plätze. Das Ergebnis zeigt, dass bekannte Plattformen, wie z.B. GoogleCode und SourceForge, nach unserem Bewertungsschema schlecht abgeschnitten haben und sich jeweils im letzten Drittel der Rangfolge befinden.

Nach der Aufstellung der Rangfolgen wird anschließend eine qualitative Bewertung der vorhandenen Werkzeuge auf den beiden besten Plattformen durchgeführt. Das exakte Vorgehen wird im folgenden Abschnitt erläutert.

6.1.6. Usability und Integration

Zunächst werden die beiden Begriffe Usability und Integration definiert, damit eine einheitliche Begriffsdefinition für die Fachstudie vorhanden ist. Danach werden Kriterien definiert, welche die Usability und Integration beschreiben. Nach diesen Kriterien werden dann anschließend die beiden besten Plattformen bewertet.

Usability

Der für die Fachstudie verwendete Begriff Usability definiert sich durch die drei folgenden Kriterien:

- **Übersicht** Die Plattform ist übersichtlich gestaltet, sodass alle Werkzeuge auf Anhieb gefunden werden können.
- **Bedienbarkeit** Die einzelnen Werkzeuge und ihre Nutzung sollen mit wenigen „Klicks“ möglich sein und einfach und intuitiv zu bedienen sein.
- **Lernbarkeit** Die einzelnen Werkzeuge erfordern keine lange Einarbeitungszeit.

Integration

Der für die Fachstudie verwendete Begriff Integration definiert sich durch die drei folgenden Kriterien:

- **Einheitlichkeit** Die vorhandenen Werkzeuge auf der Plattform sollen ein einheitliches Aussehen haben, sodass die Zusammengehörigkeit ersichtlich ist.

- **Lokalität** Die vorhandenen Werkzeuge sollen sich auf der Plattform befinden und sollen nicht auf externe Anbieter verlinkt werden.
- **Durchgängigkeit** Die Werkzeuge können untereinander Daten austauschen, z.B. in einem Wiki-Eintrag auf ein Problem verlinken.

Bewertungsschema

Die beiden besten Plattformen werden zusätzlich qualitativ bewertet. Dies geschieht auf Basis der Prüfung der vorhandenen Werkzeuge auf der Plattform bezüglich der Usability und Integration. Dieser zusätzliche Schritt ist notwendig, da bisher nur eine quantitative Prüfung der Plattformen stattgefunden hat. Durch die Bewertung der Qualität der Werkzeuge wird dem Leser ein subjektiver Eindruck der Plattform durch die Bearbeiter vermittelt, sodass dem Leser bei der Wahl der Plattform nicht nur quantitative, sondern auch qualitative Aussagen zur Verfügung stehen.

Für die Bewertung der Usability und Integration werden die festgelegten Kriterien herangezogen und anhand einer Skala eingestuft. Da Integration und Usability objektiv schwer zu bewerten sind, wird die Bewertung von zwei Personen unabhängig voneinander durchgeführt. Das endgültige Ergebnis entsteht aus dem Mittel der Ergebnisse der beiden Personen.

Je nach Ausprägung wird für jedes Kriterium ○, ◐ oder ● vergeben. Insgesamt sind für Integration und Usability jeweils maximal drei ● zu erreichen.

Trifft ein Kriterium nicht zu, z.B., dass die Werkzeuge alle unterschiedlich aussehen, wird ein ○ vergeben. Trifft ein Kriterium voll zu, z.B., dass die Werkzeuge alle gleich aussehen, wird ein ● vergeben. Trifft ein Kriterium teilweise zu, z.B., dass einige Werkzeuge gleich aussehen und einige unterschiedlich, wird ein ◐ vergeben.

6.1.7. Ergebnisse der Usability und Integration

Aus den Ergebnisse des ersten Bewertungsschrittes ist erkennbar, dass die beiden besten Plattformen für alle Nutzungsszenarien GForge und JavaForge sind. Daher wird für diese beiden Plattformen nachfolgend die Usability und Integration bewertet und in textueller und tabellarischer Form aufgelistet.

GForge

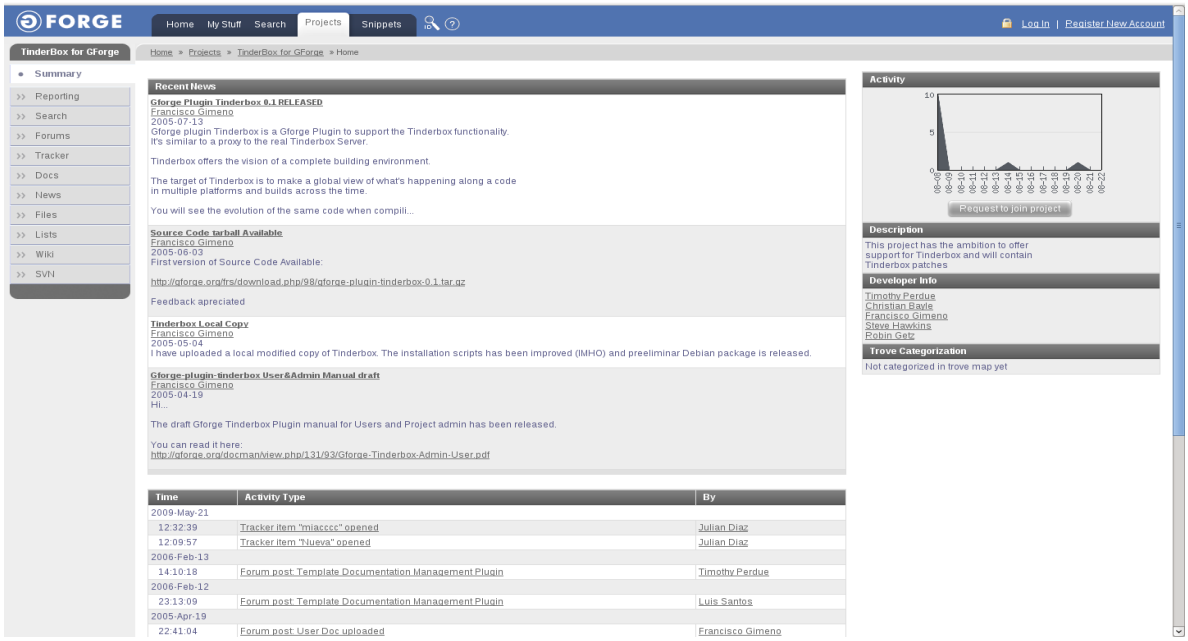


Abbildung 6.1.: Screenshot GForge

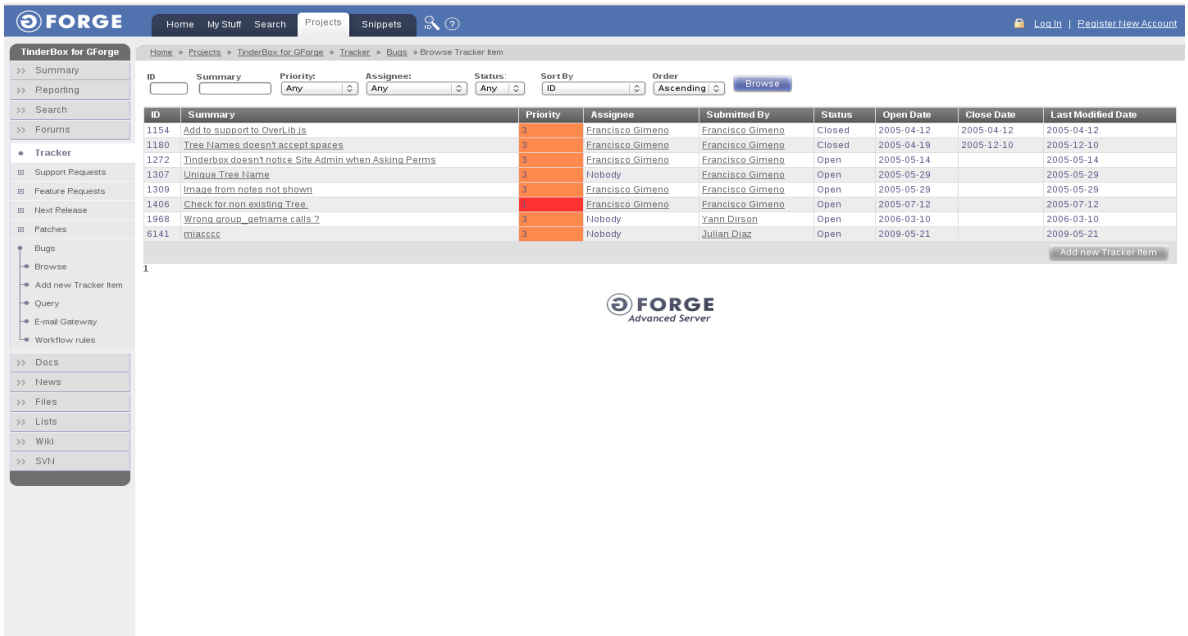


Abbildung 6.2.: Screenshot GForge Bugtracker

Bei GForge sind alle Werkzeuge auf einen Blick sichtbar und alle wichtigen Informationen sind für den Benutzer klar erkennbar. Die Werkzeuge sind alle einheitlich aufgebaut. Durch eine Menüleiste sind alle Werkzeuge aufgelistet, sodass man schnell zu den einzelnen Werkzeugen findet. Jedoch muss man sich erst einmal zurecht finden. Auf Anhieb ist es schwierig in den vielen Menüpunkten herauszufinden, wo man denn konkret Einträge erstellen kann. Ebenfalls schnell ist die Ausführung von bestimmten Tätigkeiten, so sind die Erstellung von Aufgaben oder die Erstellung eines neuen Foreneintrages schnell gemacht. Alle Werkzeuge lassen sich intuitiv und einfach bedienen, sodass man keine lange Einarbeitungszeit benötigt. Die Werkzeuge können miteinander verlinkt werden, das bedeutet, dass Wiki-Einträge auf Foreneinträge verlinken können. Alle Werkzeuge werden von GForge zur Verfügung gestellt. Es wird nicht auf externe Anbieter verlinkt.

Nachfolgend wird das Ergebnis der Bewertung der Usability und Integration aufgelistet. Dabei handelt es sich bereits um den Mittelwert der aus den Ergebnissen der beiden Bearbeiter ermittelt wurde. Die einzelnen Bewertungen sind im Anhang zu finden:

Integration		
Einheitlichkeit	Lokalität	Durchgänigkeit
●	●	●
Usability		
Übersicht	Bedienbarkeit	Lernbarkeit
◐	●	●

Tabelle 6.7.: Bewertung der Usability und Integration bei GForge

JavaForge

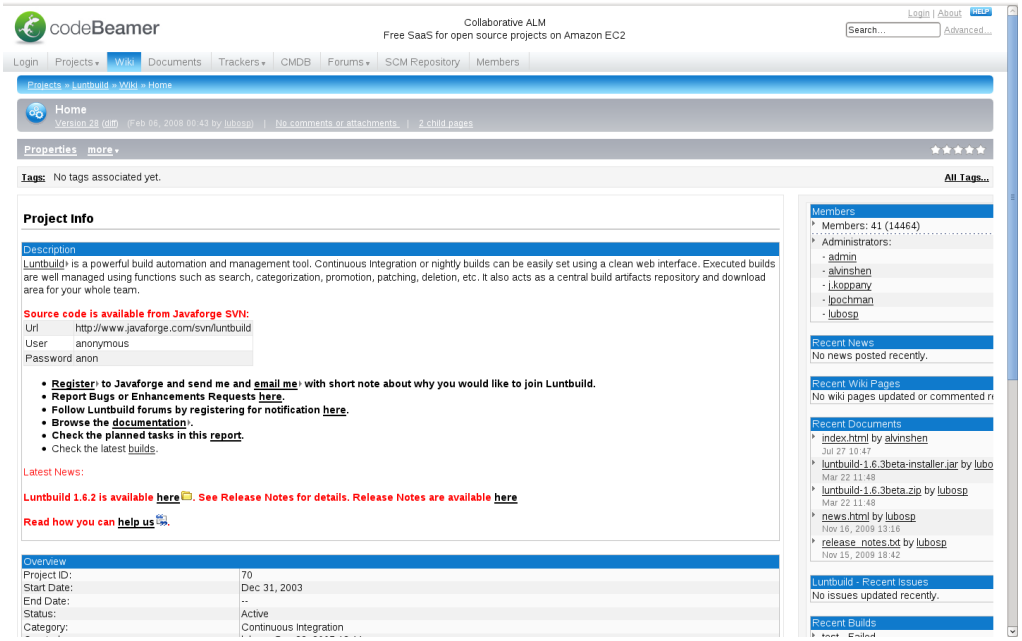


Abbildung 6.3.: Screenshot JavaForge

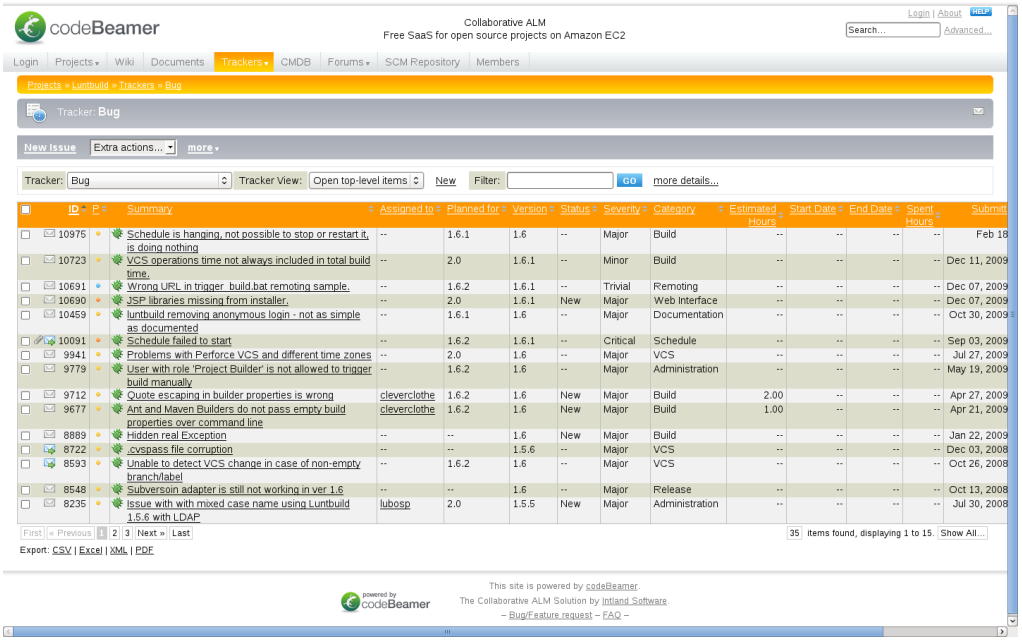


Abbildung 6.4.: Screenshot JavaForge Bugtracker

Bei JavaForge sind alle Werkzeuge auf einen Blick sichtbar und alle wichtigen Informationen sind für den Benutzer klar erkennbar. Die Werkzeuge sind alle einheitlich aufgebaut. Durch eine Menüleiste sind alle Werkzeuge aufgelistet, sodass man schnell zu den einzelnen Werkzeugen findet. Ebenfalls schnell ist die Ausführung von bestimmten Tätigkeiten, so sind die Erstellung von Aufgaben oder die Erstellung eines neuen Foreneintrages schnell gemacht. Alle Werkzeuge lassen sich intuitiv und einfach bedienen, sodass man keine lange Einarbeitungszeit benötigt. Die Werkzeuge können miteinander verlinkt werden, das bedeutet, dass Wiki-Einträge auf Foreneinträge verlinken können. Alle Werkzeuge werden von JavaForge zur Verfügung gestellt. Es wird nicht auf externe Anbieter verlinkt.

Integration		
Einheitlichkeit	Lokalität	Durchgänigkeit
●	●	●
Usability		
Übersicht	Bedienbarkeit	Lernbarkeit
●	●	●

Tabelle 6.8.: Bewertung der Usability und Integration bei JavaForge

6.2. Empfehlung

In diesem Abschnitt wird eine konkrete Empfehlung für die einzelnen Projektprofile abgegeben.

Im Laufe der Bewertung hat sich herausgestellt, dass die Plattformen GForge und JavaForge in der Rangfolge immer die ersten beiden Plätze belegt haben. Daher sind diese beiden Plattformen für den nächsten Bewertungsschritt, der Bewertung der Usability und Integration, herangezogen worden. Dieser Schritt diente dazu, die beiden Plattformen nicht nur quantitativ zu bewerten, sondern auch qualitativ. Dadurch werden die einzelnen Werkzeuge auf den Plattformen miteinander subjektiv verglichen.

Das Ergebnis der Bewertung ist, dass für die Projektprofile Ein-Person-Projekt, Software-Praktikum und Studienprojekt JavaForge die geeignetste Plattform ist.

Daher ist die Empfehlung der Fachstudie für diese drei Projektprofile JavaForge.

Für das interne Projekt belegen GForge und JavaForge ebenfalls die ersten beiden Plätze. Jedoch ist GForge hier auf Platz eins und JavaForge auf Platz zwei. Durch die qualitative Bewertung der Werkzeuge hat sich herausgestellt, dass JavaForge besser zu bedienen ist und obwohl sie die Tätigkeit Entscheidungsfindung nur teilweise unterstützt, die Empfehlung für das interne Projekt ist.

Projektprofil	Empfehlung
Ein-Person-Projekt	JavaForge
Software-Praktikum	JavaForge
Studienprojekt	JavaForge
Internes-Projekt	JavaForge

Tabelle 6.9.: Empfehlungen für die Projektklassen

6.2.1. Empfehlung für liberAda

JavaForge unterstützt alle Tätigkeiten, die für das Projekt liberAda notwendig sind. Dabei werden die Anforderungen an die Unterstützung der geforderten Tätigkeiten mehrheitlich übertroffen. Daher ist JavaForge die Empfehlung für das konkrete Projekt liberAda.

Tätigkeiten	JavaForge	liberAda	Bewertung
Problemverfolgung	●	◐	+
Dokumentenverwaltung	◐	○	+
Wissenserfassung	●	○	+
Versionskontrolle	●	◐	+
Asynchron	●	○	+
Projektüberwachung	●	◐	+
Zugriffskontrolle	●	◐	+
Projektpräsentation	◐	◐	=
Entscheidungsfindung	○	○	=
Aufgabenverwaltung	●	○	+
Zeiterfassung	●	◐	+
Aufwandsschätzung	◐	◐	=

Tabelle 6.10.: Konkrete Bewertung für liberAda

7. Fazit

In diesem Kapitel wird eine Zusammenfassung der kompletten Fachstudie erstellt. Die Arbeit wird rückblickend reflektiert und bewertet, was gut und was schlecht gelaufen ist. Zusätzlich wird ein Ausblick gegeben und ein Fazit aus der Arbeit wird gezogen.

7.1. Zusammenfassung

Das Ziel der Fachstudie war es, Empfehlungen für die Nutzung von Kollaborationsplattformen auszusprechen. Dazu wurden Projekte in Projektprofile aufgeteilt und Nutzungsszenarien mit Hilfe von Interviews von ausgewählten Projektrepräsentanten respektive Projektprofile erstellt. Aus den Nutzungsszenarien und den Erkenntnissen der Interviews wurde ein Bewertungsschema festgelegt und anhand eines vordefinierten Ablaufs jede in Betracht kommende Plattform bewertet. Um auch qualitative Aussagen zu den Plattformen machen zu können, wurden ausgewählte Plattformen in Bezug auf ihre Bedienbarkeit untersucht. Zum Schluss wurden Empfehlungen für jedes Projektprofil ausgesprochen.

7.2. Ergebnisse

JavaForge und GForge haben sich eindeutig für jedes Projektprofil als die geeignetsten Kollaborationplattform erwiesen. Beide Plattformen unterstützen die Entwickler bei den meisten in einem Softwareprojekt anfallenden Tätigkeiten. JavaForge rangiert dabei aufgrund der besseren Integration und Bedienbarkeit der Werkzeuge vor GForge.

7.3. Schwierigkeiten der Fragestellung

Die größte Herausforderung der Fachstudie bestand darin, in der inhomogenen Landschaft der Kollaborationsplattformen und Softwareprojekte eine gemeinsame Basis für eine objektive Bewertung zu finden. Was ist ein Projekt? Was zeichnet eine Kollaborationsplattform aus? Was muss sie an Funktionalität bieten und welchen Zweck muss sie erfüllen? Reicht eine quantitative Bewertung einer Plattform allein auf Grundlage der vorhandenen Werkzeuge

und wenn nein, wie kann und soll eine qualitative Bewertung durchgeführt werden? Das sind Fragen, deren Beantwortung den Autoren der Fachstudie nicht einfach erschienen. Zudem ist die Zuordnung einer Tätigkeit zu einem Werkzeug nicht eindeutig. Ein Werkzeug kann auch verwendet werden, um eine Aufgabe, für die es nicht geschaffen wurde, zu erledigen. So ist zum Beispiel eine Spezifikation für ein kleines Projekt durchaus mit einem Wiki sehr gut zu realisieren. Das Nichtvorhandensein eines Werkzeugs schließt demnach die Durchführung einer Tätigkeit auf der Plattform nicht aus.

7.4. Ausblick

Es bleibt die Frage nach der Qualität der Werkzeuge auf den Plattformen. Ausgesuchte Testfälle und klar definierte Anforderungen an die Funktionalität der Werkzeuge sind notwendig, um ausreichend verwertbare Empfehlungen für eine längere Nutzung der Plattform zu geben. Der Erfolg eines Softwareprojekts hängt empfindlich von den Arbeitsprozessen der Entwickler und den dafür verwendeten Werkzeugen ab. Eine genauere Untersuchung der Werkzeuge ist deshalb unerlässlich. Weiterhin ist die Fragestellung interessant, in wieweit die Kollaborationsplattform gute Softwareprinzipien unterstützt, indem sie beispielsweise den Entwicklern die Durchführung gewisser Arbeitsschritte vorschreibt. Ein weiterer Indikator für den Erfolg einer Kollaborationsplattform ist zudem die Kommunikation zwischen den Entwicklern. Hierfür könnte die Betrachtung von vielen großen Softwareprojekten Aufschlüsse zum Verhalten der Entwickler und die Art ihrer Kommunikation innerhalb der Plattform geben.

A. Interview-Leitfaden

Für die Durchführung der Interviews werden Interview-Partner benötigt. Damit diese Interview-Partner vorab einen Überblick über das Interview erhalten, wird ihnen ein Interview-Leitfaden zugeschickt. Dadurch haben die Interview-Partner einen ersten groben Überblick darüber, welche Fragen im Interview gestellt werden. Der folgende Leitfaden wurde ihnen zugesendet.

Universität Stuttgart

Institut für
Softwaretechnologie

Abteilung
Software Engineering

Universitätsstraße 38
70569 Stuttgart
Deutschland

Interviewleitfaden

Nutzung von Kollaborationsplattformen in Softwareprojekten

Fachstudie

Kolloborationsplattformen
zur Softwareentwicklung

Aleksandar Tolev - Modood Alvi - Timotheus Giuliani

Zweck

Dieses Dokument soll Ihnen als Leitfaden für das Interview dienen. Es werden die Fragen aufgelistet, die während des Interviews gestellt werden, damit Sie vorab einen groben Überblick bekommen. Diese Fragen dienen lediglich als Leitfaden. Während des Interviews können weitere, detaillierte Fragen gestellt werden, die hier nicht aufgeführt sind.

Einleitung

Die Entwicklung vieler Software-Produkte, speziell im Open-Source-Bereich, findet heute geographisch und organisatorisch verteilt statt: die Beteiligten - Entwickler und Benutzer - sitzen in verschiedenen Winkeln der Erde, arbeiten zu unterschiedlichen Zeiten, haben unterschiedliche Verpflichtungen und Interessen. Zur Koordinierung der Entwicklung kommen aus diesem Grund Internet-basierte Kollaborationsplattformen zum Einsatz, welche die Zusammenarbeit und Kommunikation durch Werkzeuge unterstützen sollen. Zu diesen Werkzeugen gehören etwa Konfigurationsverwaltungssysteme, Bugtracker oder Diskussionsforen; zusätzlich können die Plattformen auch genutzt werden, um Informationen über die Software oder die Möglichkeit zum Download anzubieten.

Im Rahmen einer Fachstudie der Abteilung SE an der Universität Stuttgart werden Kollaborationsplattformen miteinander verglichen und bewertet. Die Plattformen werden untersucht, um zu beurteilen für welche Art von Projekten eine Plattform geeignet ist. Dafür wird eine Bewertung abgegeben.

Wir haben Sie ausgewählt, da uns Ihr Projekt für unsere Untersuchung interessant erscheint. Die Teilnahme am Interview ist freiwillig. Wir würden uns sehr freuen, wenn Sie uns bei unserer Untersuchung unterstützen.

Die Daten, die wir in diesem Interview erheben, behandeln wir vertraulich. Ihr Daten dienen als Grundlage für Nutzungsszenarien, die wir im Rahmen der Fachstudie erstellen. Die Nutzungsszenarien ermöglichen es uns, Projekte in Kategorien zusammenzufassen.

Für Ihre Teilnahme bedanken wir uns bereits jetzt herzlich.

Inhalt des Interviews

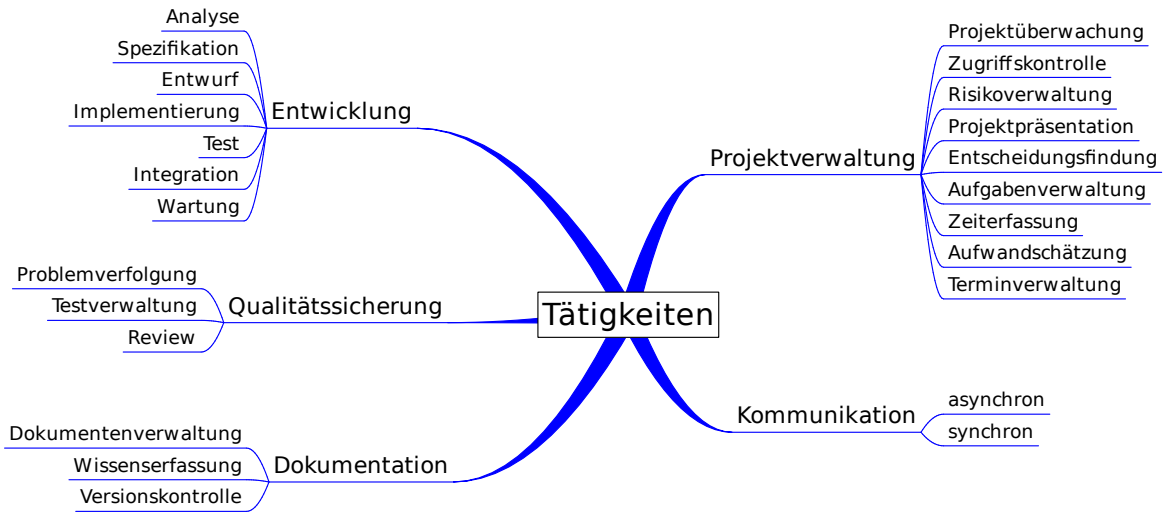
Beschreibung des Projekts

- Beschreiben Sie Ihr Projekt.
- Wie viele Entwickler arbeiten in Ihrem Projekt?

- An welchen Standorten findet die Entwicklung statt?
- Nutzen Sie eine Kollaborationsplattform wie z.B. SourceForge oder GoogleCode für Ihr Projekt? Wenn ja, nennen Sie diese bitte.

Tätigkeiten in der Entwicklung

Die nachfolgende Grafik zeigt typische Tätigkeiten im Rahmen der Softwareentwicklung. Uns interessiert, ob und wie Sie die aufgeführten Tätigkeiten in Ihrem Projekt durchführen, welche Werkzeuge Sie dafür einsetzen und ob die eingesetzten Werkzeuge in einer Kollaborationsplattform integriert sind.



Zu jeder aufgeführten Tätigkeit interessiert uns:

- Führen Sie diese Tätigkeit durch?
- Nutzen Sie hierfür Werkzeuge?
- Sind die genutzten Werkzeuge Teil der Plattform?

  Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

Sonstige Tätigkeiten

- Welche, außer die von uns genannten Tätigkeiten bei der Softwareentwicklung führen Sie in Ihrem Projekt noch aus?
- Welche Werkzeuge vermissen Sie?

Wir danken Ihnen herzlichst für Ihre Teilnahme am Interview.

B. Interview-Fragenkatalog

Der folgende Fragenkatalog wurde während eines Interviews von den Interviewern dazu verwendet, um die Fragen an den Interview-Partner zu stellen und dessen Antworten festzuhalten. Diese Ergebnisse werden anschließend verwendet, um ein Nutzungsszenario abzuleiten.

Projekttyp: _____

Projektname: _____

Projektdauer: _____

Interview-Partner: _____

Interviewer: _____

Anwesende: _____

Datum: _____

Uhrzeit: _____

Beschreibung des Projekts

1. Beschreiben Sie Ihr Projekt

   _____

2. Wie viele Entwickler arbeiten in Ihrem Projekt?

   _____

3. An welchen Standorten findet die Entwicklung statt?

   _____

4. Nutzen Sie eine Kollaborationsplattform wie z.B. SourceForge oder GoogleCode für Ihr Projekt? Wenn ja, nennen Sie diese bitte.

   _____

Tätigkeiten in der Entwicklung

Entwicklung - Analyse

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Entwicklung - Spezifikation

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Entwicklung - Entwurf

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Entwicklung - Implementierung

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Entwicklung - Test

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Entwicklung - Integration

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Entwicklung - Wartung

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Qualitätssicherung - Problemverfolgung

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Qualitätssicherung - Testverwaltung

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Qualitätssicherung - Prüfung

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Dokumentation - Dokumentenverwaltung

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Dokumentation - Wissenserfassung (Mit Beispielen erklären)

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Dokumentation - Versionskontrolle

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Projektverwaltung - Projektüberwachung

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Projektverwaltung - Zugriffskontrolle

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Projektverwaltung - Risikoverwaltung

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Projektverwaltung - Projektpräsentation

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Projektverwaltung - Entscheidungsfindung

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Projektverwaltung - Aufgabenverwaltung

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Projektverwaltung - Zeiterfassung

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Projektverwaltung - Aufwandschätzung

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Projektverwaltung - Terminverwaltung

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Kommunikation - (asynchron) z.B. E-Mail, Forum

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Kommunikation - (synchron) Chat, Telefonat

1. Führen Sie diese Tätigkeit durch? _____
2. Nutzen Sie hierfür Werkzeuge? _____
3. Sind die genutzten Werkzeuge Teil der Plattform?

   Falls nein, hätten Sie gern ein Werkzeug auf der Plattform dafür?

   _____

Sonstige Tätigkeiten

1. Welche, außer die von uns genannten Tätigkeiten bei der Softwareentwicklung führen Sie in Ihrem Projekt noch aus?

   _____

2. Welche Werkzeuge vermissen Sie?

   _____

3. Haben Sie noch weitere Anmerkungen?

   _____

C. Interview-Ergebnisse

Die folgenden Abschnitte beinhalten die detaillierten Ergebnisse der einzelnen Interviews.

C.1. Ein-Personen-Projekt

Nachfolgend werden die Ergebnisse des Interviews mit dem Ein-Personen-Projekt Liberada, welches am 13.04.2010 mit dem Entwickler Ivan Bogicevic stattgefunden hat, aufgelistet.

Projekttyp: Ein-Personen-Projekt

Projektname: LiberAda

Interview-Partner:Ivan Bogicevic

Interviewer: Aleksandar Tolev

Anwesende: Aleksandar Tolev, Ivan Bogicevic, Markus Knauss, Modood Alvi, Timo Giuliani

Datum: 13.04.2010

Uhrzeit: 16:30

Dauer: 68 Minuten

LiberAda ist ein internes Projekt am Institut ISTE der Universität Stuttgart zur Generierung von Ada Kommentaren. Diese Kommentare werden ähnlich wie in JavaDoc durch spezielle Konstrukte gekennzeichnet und dann generiert. Die generierten Kommentare werden dann als HTML exportiert. An diesem Projekt war ein Entwickler beteiligt. Die Entwicklung fand lokal statt. Für die Entwicklung wurde keine Kollaborationsplattform genutzt. Es wurde nur ein SVN Repository auf einem eigenen Server aufgesetzt.

Die Analyse wurde mit Skizzen und Zeichnungen auf einer Tafel und durchgeführt. Die Ergebnisse wurden in einer Textdatei erfasst. Der Interview-Partner hätte sich für die Analyse das Werkzeug UMLet gewünscht, welches ihn unterstützt. Die Risiken wurde nicht abgeschätzt. Die Aufwandschätzung wurde durch eine Machbarkeitsanalyse durchgeführt. Die beiden Entwickler saßen zusammen und haben diskutiert, wie lange sie ungefähr brauchen werden, um eine bestimmte Komponente zu implementieren. Die Spezifikation

wurde als Textdatei verfasst und mit Skizzen angereichert. Auf der Tafel wurden Ideen festgehalten. Die gesammelten Ideen und Komponente für den Entwurf wurden auf einer Tafel festgehalten und als OpenOffice Dokument festgehalten. Der Interview-Partner hätte sich für den Entwurf das Werkzeug UMLet gewünscht, um eine Gesamtübersicht zu erstellen. Die darauf folgende Implementierung wurde mit einem Texteditor, ComodoEdit, Eclipse mit zusätzlichen Ada Plugin durchgeführt. Der dabei entstandene Code wurde im SVN abgelegt und verwaltet. Der Interview-Partner hätte sich für die Implementierung eine komplette Entwicklungsumgebung für Ada gewünscht. Für die Test wurde das Werkzeug TG Ada Testgenerator genutzt. Der Interview-Partner hätte sich für die Test eine Art JUnit für Ada gewünscht, welches ihn unterstützt. Die Tests wurde im SVN abgelegt. Es wurde laufend integriert, sobald es Änderungen gegeben hat. Der Code wurde ausgecheckt und das Programm gestartet. Die Entwickler wurden per E-Mail über gefundene Fehler benachrichtigt. Die Fehler haben entweder Studenten oder die Entwickler selber gefunden. Dadurch wurde das Projekt gewartet. Es wurde eine Webseite erstellt, die den Quellcode und Folien enthielt. Die Folien dienten zur Erklärung der Software.

Probleme wurden nicht explizit verfolgt. Falls ein Problem aufgetaucht ist, wurde es in eine Textdatei eingetragen oder es wurde direkt besprochen und sofort behoben. Prüfungen, wie Reviews, wurden nur anfangs durchgeührt. Es wurde mehr der Code geprüft, indem man zusammen den Code durchgegangen ist und ihn auf Fehler überprüft hat. Sämtliche Dokumente wurden im SVN abgelegt. Die Wissenserfassung wurde nicht explizit durchgeführt. Es gab lediglich Folien mit Tipps für die Studenten. Der Interview-Partner hätte sich ein Wiki gewünscht, indem er Tipps für die Nutzung des Programms und das Wissen das er sonst über das Projekt erworben hatte niederschreiben könnte. Die abgelegten Dokumente und der Code wurden durch das SVN versioniert. Durch die integrierte Zugriffskontrolle des Servers hatte jeder Entwickler Zugriff auf seine Daten.

Aufgaben wurden in einer Liste eingetragen, die im SVN lag. Ein Entwickler konnte sich aus dieser Liste eine Aufgabe aussuchen, wenn er Zeit hatte und hat diese Aufgabe dann bearbeitet. Die Erfassung der Zeiten für die Bearbeitung der Aufgaben wurde nicht durchgeführt. Projektspezifische Termine wurden in einem Online-Kalender eingetragen. Entscheidungen wurden persönlich und direkt getroffen. Die Kommunikation innerhalb des Projektes fand per E-Mail, Forum und persönlich statt.

Die Projektüberwachung wurde mit einer Todo-Liste durchgeführt. In ihr stand was noch zu erledigen war. Diese Liste wurde gepflegt und ständig erweitert bzw. reduziert. Der Interview-Partner hätte gerne eine Übersicht gehabt, in der er erkennen kann, wie weit das Projekt ist und was noch alles zu erledigen ist.

Der Interview-Partner hat die Analyse, die Spezifikation, den Entwurf, die Wartung und die Problemverfolgung nicht mit Werkzeugen durchgeführt.

Der Interview-Partner war mit dem SVN sehr zufrieden und hat es häufig genutzt. Für den Editor Comodoedit hätte er sich mehr Komfort gewünscht. Mit dem Testgenerator war er

zufrieden. Der Interview-Partner hätte sich für die Integration Cruise Control gewünscht und UMLet zur Erstellung von Klassendiagrammen und anderen Zeichnungen.

C.2. Softwarepraktikum

Nachfolgend werden die Ergebnisse des Interviews mit dem Softwarepraktikum Horos, welches am 19.04.2010 mit dem Entwickler Patrick Diehl stattgefunden hat, aufgelistet.

Projekttyp: Software-Praktikum

Projektname: Horos

Interview-Partner:Patrick Diehl

Interviewer: Aleksandar Tolev

Anwesende: Aleksandar Tolev, Patrick Diehl, Markus Knauss, Modood Alvi, Timo Giuliani, Holger Röder

Datum: 19.04.2010

Uhrzeit: 10:00

Dauer: 45 Minuten

Horos ist ein Review Assistent, der die Gutachter und Moderatoren eines Reviews bei der Erstellung des Fragebogens behilflich ist. Die erstellten Fragebögen können importiert und exportiert werden und auf anhand der Fragen werden die Befunde gegenüber dem Prüfling festgestellt. An diesem Software-Praktikum waren drei Entwickler beteiligt. Dabei war ein Entwickler der Projektleiter und ein Entwickler war der Qualitätsbeauftragte. Die Entwicklung fand hauptsächlich lokal statt. Man hat sich mindestens einmal die Woche getroffen und gemeinsam gearbeitet. Zusätzliche Arbeit wurde zu Hause erledigt. Für die Entwicklung wurde die Kollaborationsplattform StudyForge genutzt. Gegen Projektende ist das Projekt zu GoogleCode umgezogen, da der Account bei StudyForge abgelaufen war.

Die Analyse wurde textuell mit OpenOffice erfasst. Die Anforderungen wurden festgehalten und im SVN-Repository abgelegt. Die Risiken wurde nicht explizit abgeschätzt. Sie wurden gemeinsam mit der Erstellung des Projektplans diskutiert und textuell erfasst. Die Aufwandschätzung wurde grob abgeschätzt. Nachdem das Projekt beendet wurde, wurden die erfassten Zeiten mit dem zu leistenden Aufwand verglichen. Die Spezifikation wurde als Latexdokument erstellt und mit Skizzen angereichert. Das Dokument wurde anschließend im SVN-Repository abgelegt. Der Entwurf wurde ebenfalls mit Hilfe von Latex erstellt. Die Klassendiagramme wurden mit Visual Paradigm erstellt. Auch das Entwurfsdokument wurde im SVN-Repository abgespeichert. Die darauf folgende Implementierung wurde mit Hilfe der Entwicklungsumgebung Eclipse und zusätzlichen Plugins durchgeführt. Der

dabei entstandene Code wurde im SVN abgelegt und verwaltet. Für die Tests wurde das Werkzeug Justus und Emma verwendet. Mit Justus wurden die Systemtests erstellt und verwaltet und mit Emma wurde die Codeüberdeckung geprüft. Die einzelnen Testfälle wurden im SVN-Repository abgelegt. Die Integration fand manuell statt. Nachdem ein Entwickler etwas implementiert hatte, wurde diese neue Codeteil aus dem Repository ausgecheckt und es wurde geprüft, ob die Anwendung läuft und das macht was spezifiziert wurde. Falls ein Fehler entdeckt wurde, wurde der jeweilige Entwickler benachrichtigt und der Fehler wurde beseitigt. Die Wartung fand nicht nicht statt. Es wurden nach Beendigung noch zwei Fehler entfernt, aber danach wurde die Anwendung nicht mehr gewartet. Das Projekt wurde durch GoogleCode nach außen präsentiert. Das dort integrierte Wiki wurde genutzt und im Downloadbereich standen die einzelnen Dateien zur Herunterladen zur Verfügung.

Probleme wurden per E-Mail verfolgt. Falls ein Problem aufgetaucht ist, wurde eine E-Mail an den jeweiligen Entwickler geschickt, mit der genauen Beschreibung. Dieser Entwickler hat dann den Fehler behoben. Der Interview-Partner hätte sich die Nutzung des integrierten Bugtrackers auf der Plattform StudyForge genutzt. Prüfungen, wie Reviews, wurden mit Hilfe eines Formulars durchgeführt. Das Formular wurde mit Word erstellt und diente den Gutachtern zur Bewertung ihrer Befunde. Der Interview-Partner hätte sich ein Werkzeug gewünscht, welches ihn bei den Prüfungen unterstützt. Sämtliche Dokumente wurden im SVN abgelegt. Die Wissenserfassung wurde nicht explizit durchgeführt. Erfasstes Wissen wurde per E-Mail ausgetauscht. Ein Wiki wäre seitens des Interview-Partners wünschenswert gewesen. Die abgelegten Dokumente und der Code wurden durch das SVN versioniert. Durch die integrierte Zugriffskontrolle des Servers hatte jeder Entwickler Zugriff auf seine Daten.

Aufgaben wurden im Projektplan-Dokument festgehalten. Einzelne Komponenten wurden den Entwickler zugewiesen und jede Komponente war zeitlich befristet. Der Interview-Partner hätte sich ein Ticketsystem gewünscht, mit dessen Hilfe er die Aufgaben strukturiert erfassen könnte. Die Erfassung der Zeiten für die Bearbeitung der Aufgaben wurde mit dem Werkzeug Fred durchgeführt. Projektspezifische Termine wurden mit Hilfe des Google Kalenders festgehalten. Eine Integration in die Plattform StudyForge wäre wünschenswert gewesen. Entscheidungen wurden persönlich und direkt getroffen. Bei den wöchentlichen Treffen wurde gemeinsam über eine Entscheidung diskutiert und dann abgestimmt. Die Kommunikation innerhalb des Projektes fand per E-Mail, Jabber und persönlich statt. Hier hätte sich der Interview-Partner eine Mailingliste gewünscht, damit er E-Mails direkt an alle Entwickler versenden kann.

Die Projektüberwachung wurde nicht explizit durchgeführt. Sie wurde mit Hilfe eines Gant-Diagramms und einem Termindriftdiagramm im Projektplan festgehalten. Der Interview-Partner hätte sich eine Fortschrittsanzeige gewünscht, die im den aktuellen Stand des Projektes anzeigt.

Der Interview-Partner hat nur das SVN von StudyForge genutzt. Andere Werkzeuge auf der Plattform wurden nicht genutzt. Es kamen externe Werkzeuge zum Einsatz.

Der Interview-Partner war mit dem SVN von StudyForge sehr zufrieden und hat es häufig genutzt. Der Interview-Partner hätte sich für die Zeiterfassung bereits einen Fred Server und einen Kalender auf StudyForge gewünscht. Außerdem war es dem Interview-Partner wichtig mit einer Plattform auch offline arbeiten zu können, sodass er zu einem späteren Zeitpunkt die Dokumente synchronisieren kann, wenn er wieder online arbeitet.

C.3. Studienprojekt

Nachfolgend werden die Ergebnisse des Interviews mit dem Studienprojekt ConfiSS, welches am 12.04.2010 mit dem Entwickler Gregor Latuske stattgefunden hat, aufgelistet.

Projekttyp: Studienprojekt

Projektname: ConfISS

Interview-Partner: Gregor Latuske (Entwickler)

Interviewer: Aleksandar Tolev

Anwesende: Aleksandar Tolev, Gregor Latuske, Markus Knauss, Modood Alvi, Timo Giuliani

Datum: 12.04.2010

Uhrzeit: 10:00

Dauer: 52 Minuten

ConfISS ist eine Open Source Webanwendung zur Verwaltung von Konferenzen. Mit dem System können wissenschaftliche Tagungen (Konferenzen), inklusive der Einreichung und Begutachtung der Einreichungen, einfach und komfortabel verwaltet werden. An diesem Projekt waren 12 Entwickler beteiligt, wobei eine Person nur für die Projektleitung zuständig war. Die Entwicklung fand verteilt statt. Die Teammitglieder haben sich wöchentlich lokal getroffen. Für die Entwicklung wurde eine interne Plattform, Trac[1], genutzt. Nachdem die Entwicklung abgeschlossen war, wurde das Projekt auf SourceForge weitergeführt.

Die Analyse wurde im Projekt mit Hilfe des integrierten Wikis durchgeführt. Der Interview-Partner hätte sich für die Analyse ein Werkzeug auf der Plattform gewünscht, welches ihn unterstützt. Währen der Analyse wurden die Risiken abgeschätzt. Eine konkrete Risikoverwaltung wurde nicht durchgeführt. Neben den Risikien wurde auch der Aufwand abgeschätzt. Die Aufwandschätzung wurde mit Hilfe vom Function Point Verfahren und COCOMO durchgeführt. Die Ergebnisse wurden in einer Exceltabelle festgehalten. Die Spezifikation wurde mit dem Werkzeug Latex erstellt. Für diese Tätigkeit hätte sich der

[1] http://trac.edgewall.org

Interview-Partner ein Werkzeug auf der Plattform gewünscht, welches ihn bei der Erstellung unterstützt. Der Entwurf wurde mit UML Werkzeugen erstellt und die Dokumentation wurde im Wiki vorgenommen. Die darauf folgende Implementierung wurde mit der Entwicklungsumgebung Eclipse[2] durchgeführt. Der dabei entstandene Code wurde im integrierten SVN abgelegt und verwaltet. Für die Tests wurde das Werkzeug Justus [3] genutzt. Die Tests wurden ebenfalls im integrierten SVN abgelegt. Durch Justus wurden die Testfälle auch verwaltet. Ein manueller Buildprozess auf einem eigenen Server führte die Integration aus. Nach Beendigung des Projektes, wurde das Projekt auf SourceForge hoch geladen und dort weiter betrieben. Mit dem integrierten Bugtracker und dem integrierten Forum, wurde die Wartung durchgeführt. Zustätzlich wurde eine Webseite erstellt, um das Projekt öffentilich zu präsentieren. Diese bietet einen Zugriff auf eine Demo-Webanwendung. Die Webseite wurde mit der SourceForge Projektseite verlinkt.

Für die Problemverfolgung wurden die integrierten Bugtracker genutzt. Prüfungen, wie Reviews, wurden mit Hilfe eines Texteditors durchgeführt. Zu den Prüfungen wurden die Gutachter per E-Mail eingeladen und die dabei entstandenen Ergebnisse wurden im Wiki festgehalten oder als Dokument im SVN abgelegt. Sämtliche Dokumente wurden entweder im SVN abgelegt oder im Wiki erfasst. Das Wiki diente auch zur Wissenserfassung. Die abgelegten Dokumente und der Code wurden durch das SVN versioniert. Durch die integrierte Zugriffskontrolle der Plattform hatten die Entwickler Zugriff auf ihre Daten.

Aufgaben wurden mit dem integrierten Ticketsystem verwaltet und verteilt. Durch Kommentare wurde die zuständigen Entwickler festgelegt. Die Erfassung der Zeiten für die Bearbeitung der Aufgaben wurde mit Fred[4] durchgeführt. Der Interview-Partner hätte sich für die Zeiterfassung ein Werkzeug auf der Plattform gewünscht, welches ihn bei der Erfassung unterstützt. Projektspezifische Termine wurden im integrierten Wiki festgehalten. Zusätzlich wurden die Termine über die Mailingliste verteilt. Entscheidungen wurden mündlich und demokratisch getroffen. Innerhalb des Projektes wurde über die Mailingliste, dem Wiki und einem Instant Messenger kommuniziert.

Die Projektüberwachung wurde nicht explizit durchgeführt. Anweisungen wurden über die Mailingliste oder über das Wiki verteilt.

Der Interview-Partner hat alle Tätigkeiten, in denen Textdokumente das Ergebnis waren, nicht mit Hilfe von Werkzeugen auf der Plattform durchgeführt.

Der Interview-Partner war mit der Plattform Trac zufrieden und hat das integrierte Wiki und SVN häufig genutzt. Das Forum auf der Plattform SourceForge empfand er als in Ordnung, mit dem Bugtracker war er jedoch weniger zufriedenstellend. Mit der Einbindungmöglichkeit

[2] http://www.eclipse.org
[3] http://justus.tigris.org/
[4] http://fred.mayastudios.com

von Plugins in SourceForge war der Interview-Partner aufgrund von Inkonsistenzen in der Grafischenobarfläche auch nicht zufrieden.

C.4. Internes-Projekt

Nachfolgend werden die Ergebnisse des Interviews mit dem internen Projekt Bauhaus, welches am 07.05.2010 mit dem Entwickler Thomas Eisenbarth stattgefunden hat, aufgelistet.

Projekttyp: Internes Projekt

Projektname: Bauhaus

Interview-Partner: Thomas Eisenbarth

Interviewer: Aleksandar Tolev

Anwesende: Aleksandar Tolev, Thomas Eisenbarth, Markus Knauss, Timo Giuliani

Datum: 07.05.2010

Uhrzeit: 17:00

Dauer: 108 Minuten

Bauhaus ist ein Forschungsprojekt der Universität Stuttgart mit dem Ziel eine Lösung zum erkennen von Software-Erosion, bei Projekten mit langer Entwicklungsdauer, zu entwickeln. Die Software sucht nach totem Code, Copy and Paste Fehlern, verstößen gegen den Entwurf und weiteren Fehlern die sich in einem langen Entwicklungsprozess einschleichen. Die Entwicklung des Projekts findet an 4 verschiedenen Orten statt, bei Axivion in Stuttgart, an der Universität von Stuttgart, in Bremen und in Australien. Insgesamt arbeiten rund 50 Personen am Projekt Bauhaus. Das Projekt wird seit 1996 entwickelt. Für die Entwicklung wird eine art Kollaborationsplattform genutzt. Es handelt sich Software namens Jira, die auf einem eigenen Server aufgesetzt ist.

Die Analyse wird in einem Wiki oder in Jira erfasst. Der Interview-Partner hätte sich für die Analyse ein einfaches Werkzeug welches ihn unterstützt gewünscht. Die Risiken werden nicht abgeschätzt. Auch die Aufwandschätzung wird nur informell druchgeführt. Hierbei geht der Entwickler nach seiner Erfahrung. Die Spezifikation wird als Issue in Jira verfasst. Der Entwurf entsteht auf deinem Witheboard das anschließend abfotografiert wird wird. Festgehalten werden die Resultate für Entwürfe die an externe Personen gehen in Word. Der Interview-Partner hatte sich für den Entwurf keine weiteren Werkzeug gewünscht. In der Implementierung steht es jedem Entwickler frei sein perönliches lieblings Werkzeug/Programmierumgebung zu nutzen. Der dabei entstandene Code wurde im SVN abgelegt und verwaltet. Beim Testen erden Regressionstests, Zufälligkeitstest mit Skripten und Unittests

durchgeführt. Der Interview-Partner hätte sich für die Test eine Werkzeugunterstützt gewünscht. Die Testergebnisse werde im SVN abgelegt. Die Integtration wird mit Make auf den 3 unterstützen Plattformen, Solaris, Windows, Linux automatisch ausgeführt. Es wurde eine Webseite erstellt, auf der das Projekt vorgestellt wird.

Probleme wurden explizit verfolgt und in Jira festgehlaten. Falls ein Problem auftaucht , wird in Jira ein Bugreport erstellt. Prüfungen werden durchgeührt. Es werden Stichproben genommen, Bauhaus selbst wird auf den Code angesetzt und es wird häufig gemeinsam Programmiert (Pairporgramming). Sämtliche Dokumente wurden im SVN abgelegt. Hierfür wurde eine spezielle Strucktur eingeführ. Für die Wissenserfassung wurde explizit ein eigenes Wiki aufgesetzt. Der Interview-Partner ist mit dieser Lösung voll zufrieden. Die abgelegten Dokumente und der Code wurden durch das SVN versioniert. Durch die integrierte Zugriffskontrolle des Servers hatte jeder Entwickler Zugriff die Daten.

Aufgaben werden in Jira erfasst. Hier kann die Aufgabe dann auch einem Verantwortlichen zugewiesen werden bzw. man kann sich eine Aufgabe zuweisen. Die Erfassung der Zeiten für die Bearbeitung der Aufgaben wurde nicht durchgeführt. Projektspezifische Termine wurden in einer EGroupware in einem Online-Kalender eingetragen. Entscheidungen wurden persönlich und oder über einen Wiki-Vote getroffen. Die Kommunikation innerhalb des Projektes findet per E-Mail, im Chat, im Telefonat und persönlich in Besprechungen statt.

Die Projektüberwachung wird mit Jira mit oder einer Todo-Liste durchgeführt. In der Liste wird festgelegt was noch zu erledigen ist. Der Interview-Partner ist mit der Übersicht von Jira zufrieden.

Der Interview-Partner hat die Analyse, die Spezifikation, den Entwurf, die Wartung nicht mit Werkzeugen durchgeführt.

Der Interview-Partner war mit dem SVN sehr zufrieden und hat es häufig genutzt. Auch Jira kommt sehr häufig zum Einsatz.

D. Plattformen

D.1. Marktübersicht

Die folgende Tabelle beinhaltet eine detaillierte Beschreibung der in der Fachstudie gefundenen Plattformen. Diese Marktübersicht zeigt, welche Plattformeigenschaften in welcher Ausprägung die Plattformen besitzen. Die Tabelle erhebt keinen Anspruch auf Vollständigkeit.

Tabelle D.1.: Marktübersicht Teil 1

Kollaborationsplattform	Alioth	Assembla	Beanstalk	BerliOS	Bettercodes	Bitbucket	BountySource	Codendi	CodePlex	CodeSpaces	dotscr.org	FireForge	Freepository	Gforge	GitHub	Gitorious	Gna.org	GNU Savannah	GoogleCode
Datenbank	○	○	○	●	○	○	○	○	○	○	Seit dem 18.10.2009 wird kein Hosting mehr angeboten (Stand: 02.06.2010)	○	○	○	○	○	○	○	○
Downloaddienst	○	○	○	○	○	○	○	○	●	◓		●	○	●	●	○	●	●	●
Skripting	○	○	○	●	○	○	○	○	○	○		○	○	○	○	○	○	○	○
VirtualHosting	○	○	○	○	○	○	○	○	○	◓		○	●	○	○	○	○	○	○
Uploaddienst	○	○	●	○	○	○	○	○	○	○		○	○	○	○	○	○	○	○
Sicherheit	●	●	●	●	○	●	○	●	●	◓		○	●	○	◓	○	○	○	○
Backup	○	○	○	○	○	○	●	○	○	◓		○	◓	○	○	○	○	○	○
Projektsuche	●	●	○	●	○	●	○	●	●	○		○	○	●	●	●	●	●	●
Klassifikation	●	●	○	●	○	○	○	●	○	○		●	○	●	●	●	○	○	○
Benutzersuche	●	●	○	●	○	●	○	●	○	○		○	○	○	○	○	●	●	○
Projektunterstützung	○	◓	●	●	○	●	○	○	●	◓		○	○	○	○	○	○	○	○
Projektneuigkeiten	○	◓	●	●	○	●	○	●	●	○		●	○	●	●	○	●	●	●
Projektseitensupport	○	○	○	○	○	○	○	○	○	○		○	○	○	○	○	○	○	○
Shell	○	○	○	●	○	○	○	●	○	○		○	○	○	●	○	○	○	○
Lizenz	○	○	○	●	○	○	○	○	●	○		○	○	○	○	○	○	○	○
Werbung	○	○	○	○	○	○	○	●	○	○		○	○	○	○	○	○	○	○
Testsystem	○	○	○	○	○	○	○	●	●	○		○	○	●	○	○	○	○	○
Buildsystem	○	○	●	○	○	○	○	●	○	○		○	○	○	○	○	○	○	○
Zugriffskontrolle	●	◓	●	●	○	●	●	●	●	◓		●	◓	●	◓	○	○	○	●
Dashboard	●	○	●	●	○	○	○	●	●	◓		○	○	●	●	○	○	○	○
Standardprojektseite	○	◓	○	●	○	○	○	○	●	◓		○	○	○	○	○	●	●	●
Projektrollen	●	◓	●	●	○	●	●	●	●	◓		○	○	●	○	○	○	○	●
Statistik	●	○	●	●	○	○	○	●	●	◓		○	○	●	●	○	○	○	○

Kollaborationsplattform	Kalender	Umfrage	Aktivitätshistorie	Diary	Webspace	Entwicklerprofil	Zeiterfassung	TaskTracker	Ticketsystem	Bewertung	Review	Testverwaltung	Bugtracker	Help	Mailingliste	Nachrichtenfunktion	InstantMessaging	Forum	RSS	Dokumentenmanager	Wiki	Versionskontrolle	Offline Version
Alioth	●	●	●	●	○	●	○	●	●	○	○	○	●	○	●	○	○	●	●	●	●	●	●
Assembla	○	○	◐	○	●	●	○	◐	◐	○	○	○	○	●	○	○	◐	○	◐	◐	◐	●	○
Beanstalk	○	○	●	○	○	●	○	●	●	○	●	○	●	●	○	○	●	○	●	●	○	●	○
BerliOS	○	●	●	●	●	●	○	●	●	○	○	○	●	●	●	○	○	●	●	●	○	●	○
Bettercodes	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Bitbucket	○	○	●	○	○	●	○	○	●	○	○	○	●	●	○	○	○	○	●	○	●	●	○
BountySource	○	○	●	○	○	○	○	●	●	○	○	○	●	●	○	○	○	○	○	●	●	●	○
Codendi	○	●	●	○	○	●	○	●	●	○	○	●	●	●	●	○	●	●	●	●	●	●	●
CodePlex	○	○	●	○	○	○	○	●	●	○	○	○	●	○	●	○	○	●	●	○	●	●	○
CodeSpaces	○	○	◐	○	◐	○	○	◐	◐	○	○	○	◐	○	○	○	○	○	○	◐	◐	◐	○
dotscr.org	Seit dem 18.10.2009 wird kein Hosting mehr angeboten (Stand: 02.06.2010)																						
FireForge	○	○	●	●	○	○	○	●	●	○	○	○	●	○	●	○	○	○	○	○	○	●	○
Freepository	○	○	○	○	◐	○	○	○	○	○	○	○	◐	○	○	○	○	○	◐	○	◐	●	○
Gforge	○	○	●	○	○	○	○	●	●	○	○	○	●	○	●	○	○	●	○	●	●	●	●
GitHub	○	○	●	○	○	●	○	○	○	○	○	○	●	○	○	○	○	○	○	○	●	●	○
Gitorious	○	○	●	○	○	●	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	●	○
Gna.org	○	○	●	○	○	●	○	●	●	○	○	○	●	●	○	○	○	○	○	○	○	●	○
GNU Savannah	○	○	●	○	○	●	○	●	●	○	○	○	●	●	●	○	○	○	●	○	○	●	○
GoogleCode	○	○	●	○	○	○	○	○	●	○	○	○	●	○	●	○	○	○	●	○	●	●	○

Tabelle D.2.: Marktübersicht Teil 2

Kollaborationsplattform	GoPlan	GridyZone	Huddle	icculus.org	java.net	JavaForge	Jira	KnowledgeForge	Launchpad	LibreSource	LuaForge	MacOS	Mozdev	Novell Forge	OpenSVN	Origo	OSOR.eu	Ourproject	Project Kenai
Datenbank	○	Seit wurde geschlossen (Stand: 06.06.2010)	○	○	○	○	○	○	○	○	○	○	○	Plattform nicht erreichbar (Stand: 06.05.2010)	○	○	○	●	○
Downloaddienst	◓		●	●	●	●	○	○	●	●	●	○	●		○	●	●	●	●
Skripting	○		○	●	○	○	○	○	●	○	○	○	○		○	○	○	○	○
VirtualHosting	◓		○	○	○	○	○	○	○	○	○	○	○		○	○	●	○	○
Uploaddienst	◓		●	○	○	○	○	○	●	●	○	○	○		○	○	○	●	○
Sicherheit	◓		◓	●	○	○	○	○	●	●	○	○	○		○	○	○	○	○
Backup	◓		○	○	○	○	○	○	○	○	○	○	○		○	○	○	●	○
Projektsuche	○		●	●	●	●	○	●	●	●	○	●	●		○	●	●	●	●
Klassifikation	○		○	○	●	○	○	○	○	○	○	○	○		○	○	○	●	○
Benutzersuche	○		○	○	○	○	○	●	●	●	○	○	○		○	○	●	●	●
Projektunterstützung	◓		○	○	○	◓	◓	○	○	●	●	●	○		○	○	○	○	○
Projektneuigkeiten	◓		●	○	◓	●	◓	○	●	●	○	●	●		○	●	●	●	○
Projektseitensupport	○		○	○	○	◓	○	○	●		○	●	●		○	●	○	○	●
Shell	○		○	●	○	○	○	○	○	○	○	○	○		○	○	○	○	○
Lizenz	○		○	○	●	○	○	○	●	○	●	○	○		○	○	●	●	●
Werbung	○		○	○	○	○	○	○	○	○	○	○	○		○	○	○	○	○
Testsystem	○		○	○	○	◓	○	○	○	○	○	○	○		○	○	○	○	○
Buildsystem	○		○	○	○	○	○	○	○	●	○	○	○		○	●	○	○	○
Zugriffskontrolle	◓		●	●	●	●	◓	●	●	●	●	●	●		○	●	●	●	●
Dashboard	◓		●	○	◓	◓	◓	○	●	○	○	●	●		○	●	●	○	○
Standardprojektseite	○		○	○	●	●	○	○	●	●	●	●	●		○	●	●	●	●
Projektrollen	○		●	○	●	●	◓	○	○	○	○	○	●		○	○	●	●	●
Statistik	○		○	○	●	●	◓	○	●	●	●	●	●		○	●	●	●	○

Tabelle D.3: Marktübersicht Teil 3

Kollaborationsplattform	Kalender	Umfrage	Aktivitätshistorie	Diary	Webspace	Entwicklerprofil	Zeiterfassung	TaskTracker	Ticketsystem	Bewertung	Review	Testverwaltung	Bugtracker	Help	Mailingliste	Nachrichtenfunktion	InstantMessaging	Forum	RSS	Dokumentenmanager	Wiki	Versionskontrolle	Offline Version
GoPlan	◐	○	◐	○	◐	○	◐	◐	◐	○	○	○	◐	○	○	◐	○	○	◐	◐	○	○	
GridyZone	Seit wurde geschlossen (Stand: 06.06.2010)																						
Huddle	●	○	●	○	◐	●	○	●	○	○	◐	○	○	○	○	●	○	○	●	●	●	◐	○
icculus.org	○	○	○	○	●	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	●	○
java.net	○	○	○	○	○	○	○	●	●	○	○	○	●	○	○	●	○	●	●	●	●	●	○
JavaForge	○	◐	●	●	○	○	○	●	●	○	○	○	●	○	○	○	●	●	○	●	●	●	●
Jira	◐	○	◐	○	○	○	◐	◐	◐	◐	○	○	◐	○	○	○	○	○	○	○	○	○	◐
KnowledgeForge	○	○	○	○	○	○	○	●	●	○	○	○	●	○	●	○	○	○	○	○	●	●	●
Launchpad	○	○	●	○	○	●	○	●	○	●	●	○	●	○	●	○	○	○	●	●	○	●	○
LibreSource	○	●	○	○	○	●	○	○	○	○	○	○	●	○	●	●	●	●	●	○	●	●	○
LuaForge	○	●	●	●	○	○	○	●	○	○	○	○	○	○	○	●	○	●	○	○	○	●	○
MacOS	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	●	●	○
Mozdev	○	○	●	○	○	○	○	○	○	○	○	○	●	○	●	●	●	●	●	○	●	●	○
Novell Forge	Plattform nicht erreichbar (Stand: 06.05.2010)																						
OpenSVN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●	○
Origo	○	○	○	○	○	●	○	○	●	○	○	○	○	○	○	○	○	●	●	○	●	●	○
OSOR.eu	○	○	○	●	○	●	○	○	○	○	○	○	●	○	○	●	○	●	●	●	○	●	○
Ourproject	○	●	○	●	○	●	○	●	●	○	○	○	○	○	●	○	○	●	○	●	●	●	○
Project Kenai	○	○	○	○	○	○	○	○	○	○	○	○	●	○	●	○	●	●	○	○	●	●	○

Tabelle D.4.: Marktübersicht Teil 4

Tabelle D.5.: Marktübersicht Teil 5

Kollaborationsplattform	RubyForge	Sarovar	ShareSource	SourceForge	Springloops	tigris.org	Trac	TuxFamily	unfuddle	Xp-dev
Datenbank	●	●	○	○	○	○	○	●	○	○
Downloaddienst	●	●	●	●	○	●	○	○	○	○
Skripting	○	○	○	○	○	○	○	○	○	○
VirtualHosting	○	○	○	○	○	○	○	●	●	◓
Uploaddienst	●	●	○	○	◓	○	○	●	○	○
Sicherheit	○	○	○	○	◓	○	○	●	◓	◓
Backup	●	●	○	○	●	○	○	○	○	◓
Projektsuche	●	●	●	●	○	●	○	●	○	○
Klassifikation	●	●	●	●	○	●	○	○	○	○
Benutzersuche	●	●	○	○	○	○	○	●	○	○
Projektunterstützung	○	○	○	○	●	○	●	○	●	●
Projektneuigkeiten	●	●	○	●	●	◓	●	○	●	●
Projektseitensupport	○	○	○	●	○	○	○	○	○	○
Shell	○	○	○	●	○	○	○	○	○	○
Lizenz	●	●	○	●	○	●	●	○	○	○
Werbung	○	○	○	○	○	○	○	○	○	○
Testsystem	○	○	○	○	●	○	○	○	○	○
Buildsystem	○	○	○	○	○	○	○	○	○	○
Zugriffskontrolle	●	●	○	●	●	●	●	○	●	●
Dashboard	○	○	○	○	◓	○	○	○	●	○
Standardprojektseite	●	●	●	●	●	●	○	●	●	●
Projektrollen	●	●	●	●	●	●	○	○	●	●
Statistik	●	●	○	●	○	●	○	○	○	○

Kollaborationsplattform	Kalender	Umfrage	Aktivitätshistorie	Diary	Webspace	Entwicklerprofil	Zeiterfassung	TaskTracker	Ticketsystem	Bewertung	Review	Testverwaltung	Bugtracker	Help	Mailingliste	Nachrichtenfunktion	InstantMessaging	Forum	RSS	Dokumentenmanager	Wiki	Versionskontrolle	Offline Version
RubyForge	○	●	●	●	○	●	○	●	●	○	○	○	○	○	●	○	○	●	○	●	●	●	○
Sarovar	○	●	○	●	○	●	○	●	●	○	○	○	○	○	●	○	○	●	○	●	○	●	○
ShareSource	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	●	○	●	○
SourceForge	◒	○	●	●	○	●	○	●	●	●	○	○	●	●	●	○	○	●	○	○	●	●	○
Springloops	○	○	●	○	◒	○	○	●	●	○	◒	◒	●	○	○	○	○	○	●	○	○	●	◒
tigris.org	○	○	○	○	○	○	○	●	●	○	○	○	●	○	○	●	○	●	●	●	●	●	○
Trac	◒	○	●	○	○	○	○	●	●	○	○	○	●	●	○	○	○	○	○	○	●	●	●
TuxFamily	○	○	○	○	○	○	○	○	○	○	○	○	○	●	●	○	○	○	○	○	●	●	○
unfuddle	●	○	●	○	◒	○	◒	●	●	○	○	○	●	○	○	●	○	○	○	○	●	●	○
Xp-dev	○	○	●	●	◒	○	○	●	●	○	○	○	●	○	○	○	○	●	○	○	●	●	○

Tabelle D.6.: Marktübersicht Teil 6

D.2. K.O-Kriterien

In Tabelle D.7 werden die Plattformen aufgelistet, die nicht den K.O.-Kriterien genügen.

Name	**K.O.-Kriterium**
ActiveState	Generalität
Alioth	Generalität
Apache	Werkzeuganzahl
Betavine	Generalität
Bettercodes	Verfügbarkeit
BountySource	Werkzeuganzahl
Codendi	Verfügbarkeit
dotsrc.org	Werkzeuganzahl
Drupal	Werkzeuganzahl
FireForge	Generalität
Freepository	Werkzeuganzahl
GitHub	Verfügbarkeit
icculus	Werkzeuganzahl
Java.net	Generalität
KnowledgeForge	Verfügbarkeit
LibreSource	Verfügbarkeit
LuaForge	Generalität
MacOSForge	Generalität
mozdev	Generalität
OpenSuseBuildService	Generalität
OpenSVN	Werkzeuganzahl
OSOR.eu	Generalität
OSP	Generalität
OW2Consortium	Generalität
Pgfoundry	Generalität
Project Kenai	Verfügbarkeit
RubyForge	Generalität
SimpleEndUserLinux	Generalität
TheFreeCountry	Werkzeuganzahl
tigris.org	Generalität
tuxfamily	Werkzeuganzahl
xp-dev	Werkzeuganzahl

Tabelle D.7.: Aussortierte Plattformen

D.3. Vorgehensmodell

Anhand dieser Anleitung werden die einzelnen Plattformen bewertet. Die Anleitung beschreibt, welche Daten verwendet werden, mit denen die einzelnen Tätigkeiten auf den Plattformen durchgeführt werden. Dadurch wird auf jeder Plattform das gleiche Szenario durchgespielt und es ist möglich eine einheitliche Bewertung abzugeben.

Anhand dieser Anleitung wird ein Szenario durchgespielt, um eine einheitliche Bewertung der Plattformen zu ermöglichen.

1. Erstelle ein Dokument mit dem Titel Analyse. Das Dokument soll als neu eingestuft werden und wenn möglich einem Bearbeiter zugeordnet werden. Der Inhalt der Analyse besteht aus zwei Punkten: funktionale Anforderung und nicht funktionale Anforderung. Der Punkt funktionale Anforderung hat als Unterpunkt: Speicherplatz < 500 kB. Der Punkt nicht funktionale Anforderung hat als Unterpunkt: Web 2.0 Design. Der Punkt funktionale Anforderung mit seinem Unterpunkt wird höher priorisiert als der Punkt nicht funktionale Anforderung.
2. Erstelle ein Dokument mit dem Titel Spezifikation. Das Dokument soll als neu eingestuft werden und wenn möglich einem Bearbeiter zugeordnet werden. Der Inhalt der Spezifikation lautet (Ausschnitt aus einer Beispiel Spezifikation, Kapitel 2):

2 Nichtfunktionale Anforderungen

2.1 Mengengerüst Das Programm ist eine Einzelplatzanwendung ohne Netzwerkfunktionalität und wird somit zu jedem Zeitpunkt nur von einem Anwender bedient. Das Programm soll auf jeden Fall mindes- tens 1000 Testfälle und 25 eingetragene Benutzer ohne spürbare Beeinträchtigung der Performan verwalten können.

2.2 Entwurfseinschränkungen

2.2.1 Systemumgebung Außer dem Java Runtime Environment ab Version 1.5 und der einzubindenden Bibliothek iText soll das Programm keine Abhängigkeiten haben. Zur Anzeige von PDF-Dokumenten muss der Benutzer einen PDF-Betrachter haben. Sollte das Programm nicht in der Lage sein, den systemspezifischen PDF-Betrachter zu starten, wird eine entsprechende Meldung angezeigt. Das Programm verwendet für verschiedene Funktionen einen Web Browser. Sollte das Programm nicht in der Lage sein, einen Web Browser auf dem System zu finden oder eine Website zu öffnen, wird eine entsprechende Meldung angezeigt.

2.2.2 Layout und Gestaltung Das Programm wird auf Systemen verwendet, die mindestens die Auflösung 1024x768 Pixel haben. Alle Fenster sollen, sofern sinnvoll, vergrößert und verkleinert werden können. Der Test-durchführungsassistent (später beschrieben) soll in einem m öglichst kleinen Fenster verwendet werden können, da parallel zu dem zu entwickelnden Programm ein Prüfling ausgeführt wird und beide gleichzeitig auf dem Bildschirm Platz finden sollten. Das Programm verwendet das systemnative Look & Feel von Java und wird primär für Microsoft Windows, Apple Mac OS X sowie gängige Linux-Oberflächen angepasst, um Benutzern dieser Systeme eine möglichst gute Bedienung des Programms zu ermöglichen.

2.2.3 Datenhaltung Alle Projektdaten werden in einer Arbeitsdatei in einem eigenen Datenformat gespeichert. Diese Arbeitsdatei enthält Benutzer, Testsequenzen, Testfälle und Testprotokolle. Es können prinzipiell beliebig viele Arbeitsdateien angelegt werden, jedoch kann zu jedem Zeitpunkt nur eine Arbeitsdatei offen sein. Zudem soll die Möglichkeit bestehen, Testdaten zu importieren und zu exportieren. Hierbei wird ein vom Kunden vorgegebenes XML-Datenformat verwendet. Sonstige Programmeinstellungen (Arbeitsplatzkonfiguration) werden in einer oder mehreren nicht vom Benutzer festlegbaren Konfigurationsdatei abgelegt. Zur Vorbeugung von Datenverlusten wird beim Speichern der Arbeitsdatei zunächst eine Sicherungskopie der bisherigen Arbeitsdatei erstellt und die Arbeitsdatei erst danach überschrieben.

2.3 Robustheit Das Programm soll defensiv implementiert werden, jedoch sollen die Eingaben nicht auf logische Konsistenz uberprüft werden, da als Anwender ein Softwareentwickler angenommen wird. Somit können Grundkenntnisse in der Bedienung und Anwendungsdomäne angenommen werden. Mutwillige Versuche, das Programm zum Absturz zu bringen, können ausgeschlossen werden. Programmfehler sollen jedoch nicht die Integrität der gespeicherten Daten gefährden können.

2.4 Portabilität Das zu entwickelnde Programm soll auf allen Computersystemen lauffähig sein, auf denen eine Java Runtime Environment (der Version 1.5 oder höher) mit angemessener Performanz läuft.

2.5 Erweiterbarkeit Das Programm braucht keine besonderen Funktionen zur einfachen Erweiterbarkeit bereitstellen. Es sind derzeit keine konkreten Erweiterungspläne vorhanden.

2.6 Distributionsform und Installation Das Programm wird für alle Betriebssysteme mit Java Virtual Machine als installationsfreies Java Archive (JAR) ausgeliefert. Zudem werden ggf. f ür einzelne Betriebssysteme spezielle Programmpakete bereitgestellt, um die Installation zu erleichtern.

Neben dem Text wird noch ein Use-Case Diagramm erstellt, welches die Beziehung zwischen dem System und dem Benutzer darstellt.

3. Erstelle ein Dokument mit dem Titel Entwurf. Das Dokument soll als neu eingestuft werden und wenn möglich einem Bearbeiter zugeordnet werden. Der Inhalt des Entwurfs lautet (Auszug aus einem Beispiel Entwurf, Kapitel 3):

3 Beschreibungen einzelner Programmkomponenten

3.1 Das Datenmodell (Model)

3.1.1 Die Klasse Runtime Runtime bietet einen globalen Anlaufpunkt für Programmeinstellungen, plattformspezifische Einstellungen und alle Textausgaben der Programmoberfläche. Klassen, die diese Funktionen bereitstellen, implementieren entsprechende Interfaces. Runtime selbst implementiert ebenfalls diese Interfaces und delegiert jeden Aufruf an jene konkreten Klassen.

3.1.2 Die Klasse Project Über die statische Klasse ActiveProject ist die aktive Instanz der Klasse Project ansprechbar. Prinzipiell kann es mehrere Instanzen der Klasse Project gleichzeitig geben, der Benutzer arbeitet aber immer nur mit der aktiven Instanz. Project ist ein Delegate für die Methoden des UserManager (s.u.). Zusätzlich enthält sie die für den Benutzer nicht sichtbare Wurzeltestsequenz und Wurzeltestsequenzarchiv, die alle weiteren Projektdaten enthalten.

Nebem dem Text wird noch ein UML-Diagramm (Aktivitäts-, Zustands-, Klassen-, Komponenten-, Sequenzdiagramm etc.) erstellt.

4. Erstelle eine neue Klasse und implementiere folgende Funktion in der jeweils unterstützten Sprache:

```
FOR i:=0 TO 100 DO
PRINT("Anzahl: " + i);
END;
```

Der Code wird anschließend ausgeführt. Das Ergebnis wird in der Konsole ausgegeben. Danach wird ein Breakpoint in der PRINT Anweisung gesetzt und die Funktion wird auf Fehler überprüft, indem die einzelnen Schleifendurchläufe auf Korrektheit überprüft werden.

5. Erstelle einen Testfall für den oben erwähnten Code, der jeden Schleifendurchlauf auf Korrektheit überprüft. Der Testfall wird auf der Plattform nach einer vorgegebenen Zeit automatisch ausgeführt. Betrachte die Testergebnisse und überprüfe sie auf Korrektheit.
6. Passe die Konfiguration für das Buildsystem an. Das Projekt soll täglich um 00:00 Uhr komplett ausgecheckt und gebaut werden. Das Bauen des Systems läuft dann zum festgelegten Zeitpunkt automatisch ab und benachrichtigt die Entwickler, ob es Fehler gegeben hat oder ob das Bauen erfolgreich war.
7. Verändere die fertig gestellte Software und speichere sie zentral ab.
8. Erstelle einen Eintrag, in dem ein gefundener Fehler des Systems festgehalten wird. Diese Fehlermeldung wird in einer einheitlichen Übersicht dargestellt. Der Fehler bekommt eine hohe Priorität und wird einem Bearbeiter zugeordnet.
9. Die erstellten Tests werden in einer Übersicht dargestellt. Wähle einen Testfall aus und bearbeite ihn. Wähle einen weiteren Testfall aus und lösche ihn.
10. Wähle ein Dokument aus, das geprüft werden soll und markiere es als Prüfling. Weise weiteren Entwicklern die Aufgabe zu, dass sie an dieser Prüfung teilnehmen und welche Rollen sie haben. Fasse die gefundenen Fehler als Kommentare zusammen und verlinke sie mit dem Prüfling, sodass auf einem Blick die gefundenen Fehler sichtbar sind. Setze den Status des Prüflings nach der Prüfung, je nachdem wie schwerwiegend die Fehler sind.
11. Die erstellten Dokumente werden in einer Übersicht dargestellt. Wähle ein Dokument aus und bearbeite es. Wähle ein weiteres Dokument aus und lösche es.
12. Erstelle einen Eintrag im Wiki, der als Titel Eclipse SVN Konfiguration trägt und folgenden Inhalt besitzt:

    Subclipse Eclipse Update Site: `http://subclipse.tigris.org/update_1.4.x`. Man benötigt mindestens das Subclipse und das Java HL Adapter Paket.

    Der Eintrag soll auf der Hauptseite verlinkt werden, damit Benutzer direkt auf diesen Eintrag kommen.
13. Erstelle ein Dokument mit dem Titel Versionskontrolle. Füge das Dokument der Versionskontrolle hinzu. Überprüfe die Version der des Dokuments. Lade das Dokument herunter und füge eine neue Zeile hinzu:

Das ist ein Test für die Versionskontrolle

Lade das Dokument erneut hoch und überprüfe die Version. Diese muss sich nun um eine Version erhöht haben.

14. Erstelle einen Eintrag im Forum und antworte darauf. Erstelle eine neue Nachricht in der Nachrichtenfunktion. Versende über die Mailingliste eine E-Mail. Führe eine Unterhaltung mit dem Chatprogramm.
15. In einer Übersicht werden alle Statistiken und Daten zum Projekt angezeigt. Die bevorstehenden Termine, die zu erledigenden Aufgaben, der Entwicklungsverlauf und weitere Informationen werden angezeigt.
16. Logge dich mit deinem registrierten Benutzernamen und deinem Passwort ein. Die eigene persönliche Seite wird angezeigt und man hat nur Zugriff auf die eigenen Dateien.
17. Wähle in einer Übersicht ein Risiko aus. Es werden Details zum Risiko angezeigt. Dazu gehören die Beschreibung des Risikos, ob das Risiko schon eingetroffen ist und wie hoch die Wahrscheinlichkeit des Risikos ist.
18. Erstelle eine Seite, in der das Projekt präsentiert werden kann. Lade Dateien hoch, die Benutzer herunterladen können und erstelle eine Beschreibung, die das Projekt charakterisiert. Erstelle eine Rubrik, in der häufig gestellte Fragen beantwortet werden.
19. Erstelle eine Frage, bei gemeinsam abgestimmt werden soll. Benachrichtige die Entwickler, die an der Abstimmung teilnehmen sollen, damit sie ihre Stimme abgeben.
20. Erstelle eine neue Aufgabe mit dem Titel Testaufgabe. Weise diese Aufgabe einem Entwickler zu und markiere sie als sehr wichtig. Der Entwickler, der die Aufgabe bearbeiten soll, kann sich die Aufgabe in einer Übersicht anschauen und bearbeiten. Markiere die Aufgabe als fetig gestellt.
21. Erfasse die benötigte Zeit für eine Aufgabe udn lege den Zeitraum fest, in dem gearbeitet wurde. Speichere die erfasste Zeit ab.
22. Erstelle eine Aufwandschätzung für eine Aufgabe und schätze den Aufwand auf 20 Stunden. In einer Übersicht werden alle Aufwände angezeigt.
23. Erstelle einen Termin mit dem Titel Konferenz. Die Konferenz fängt am 25.05.2010 um 10:00 Uhr an und endet am 25.05.2010 um 12:00 Uhr. Weise dem Termin Entwickler zu, die daran teilnehmen werden. Benachrichtige die Teilnehmer über den Termin.

E. Bewertung

In diesem Kapitel werden die detaillierten Bewertungen der einzelnen Plattform für die jeweiligen Nutzungsszenarien aufgelistet.

E.1. Ein-Personen-Projekt

	Assembla	BerliOs	Bitbucket	CodePlex	CodeSpaces	Gforge	Gitorious	Gna.org	GNU Savan	GoogleCode	GoPlan	Huddle	JavaForge	Launchpad	Origo	Ourproject	Sarovar	SourceForge	Unfuddle
Problemverfolgung	=	+	=	=	+	+	+	+	+	+	=	=	+	+	+	+	+	+	=
Dokumentenverwaltung	+	+	+	=	+	+	+	+	+	+	+	+	+	+	+	+	+	=	+
Wissenserfassung	+	+	+	+	+	+	+	+	+	+	=	+	+	+	+	+	=	+	+
Versionskontrolle	+	+	+	+	+	+	+	+	+	+	-	=	+	+	+	+	+	+	+
Asynchron	+	+	+	+	+	+	=	+	+	=	+	+	+	+	+	+	+	+	+
Projektüberwachung	+	+	-	=	+	+	=	+	+	=	=	=	+	+	+	=	=	+	=
Zugriffskontrolle	=	+	+	+	=	+	+	+	+	+	=	+	+	=	=	+	+	=	+
Projektpräsentation	+	=	-	=	-	+	=	=	=	=	-	=	=	=	=	=	=	=	=
Entscheidungsfindung	=	+	=	+	=	=	=	+	+	=	=	+	=	+	=	+	+	=	+
Aufgabenverwaltung	+	=	=	+	+	+	=	+	+	=	+	+	+	+	+	+	+	+	+
Zeiterfassung	+	-	-	-	+	+	-	-	-	-	+	-	+	-	-	-	-	-	-
Aufwandsschätzung	-	-	-	-	+	=	-	-	-	-	-	=	=	=	-	-	-	-	-

Tabelle E.1.: Detaillierte Bewertung für das Ein-Personen-Projekt

E.2. Software-Praktikum

	Assembla	BerliOs	Bitbucket	CodePlex	CodeSpaces	Gforge	Gitorious	Gna.org	GNU Savan	GoogleCode	GoPlan	Huddle	JavaForge	Launchpad	Origo	Ourproject	Sarovar	SourceForge	Unfuddle
Problemverfolgung	=	+	=	=	+	+	+	+	+	+	=	=	+	+	+	+	+	+	=
Dokumentenverwaltung	=	+	=	=	+	=	=	+	+	=	=	+	=	+	+	+	+	-	+
Wissenserfassung	+	+	+	+	+	+	+	+	+	+	=	+	+	+	+	+	=	+	+
Versionskontrolle	=	=	=	+	+	=	=	=	=	=	- -	-	=	=	=	=	=	=	=
Asynchron	+	+	=	+	+	+	-	=	=	-	+	+	+	+	+	+	+	+	+
Projektüberwachung	+	+	-	-	+	+	=	+	+	=	=	=	+	+	+	=	=	+	=
Zugriffskontrolle	=	+	+	+	=	+	+	+	+	+	=	+	+	=	=	+	+	=	+
Projektpräsentation	+	=	-	-	-	+	=	=	=	=	-	=	=	=	=	=	=	=	=
Entscheidungsfindung	-	+	-	-	-	-	-	+	+	-	-	=	-	=	-	+	+	-	=
Aufgabenverwaltung	+	+	-	-	+	+	-	+	+	-	+	+	+	-	+	+	+	=	+
Zeiterfassung	=	- -	- -	- -	=	=	- -	- -	- -	- -	=	- -	=	- -	- -	- -	- -	- -	- -
Aufwandsschätzung	-	-	-	-	+	=	-	-	-	-	-	=	=	-	-	-	-	-	-

Tabelle E.2.: Detaillierte Bewertung für das Software-Praktikum

E.3. Studienprojekt

	Assembla	BerliOs	Bitbucket	CodePlex	CodeSpaces	Gforge	Gitorious	Gna.org	GNU Savan	GoogleCode	GoPlan	Huddle	JavaForge	Launchpad	Origo	Ourproject	Sarovar	SourceForge	Unfuddle
Problemverfolgung	=	+	=	=	+	+	+	+	+	+	=	=	+	+	+	+	+	+	=
Dokumentenverwaltung	=	+	=	-	=	=	=	+	+	=	=	+	=	+	+	+	+	-	+
Wissenserfassung	+	=	+	=	=	=	=	-	-	=	- -	=	=	=	=	=	- -	=	=
Versionskontrolle	+	=	+	=	=	=	=	=	=	=	- -	-	=	=	=	=	=	=	=
Asynchron	+	+	=	+	+	+	-	=	=	-	+	+	+	+	+	+	=	+	+
Projektüberwachung	+	+	-	=	+	+	=	+	+	=	=	=	+	+	+	=	=	+	=
Zugriffskontrolle	-	=	=	=	-	=	=	=	=	=	-	=	=	-	-	=	=	-	=
Projektpräsentation	+	=	-	=	-	+	=	=	=	=	-	=	=	=	=	=	=	=	=
Entscheidungsfindung	-	+	-	=	-	-	-	+	+	-	-	=	-	=	-	+	+	-	=
Aufgabenverwaltung	=	=	- -	=	=	=	- -	=	=	- -	=	=	=	-	=	=	=	-	=
Zeiterfassung	=	- -	- -	- -	=	=	- -	- -	- -	- -	=	- -	=	- -	- -	- -	- -	- -	- -
Aufwandsschätzung	-	-	-	-	+	=	-	-	-	-	-	=	=	=	-	-	-	- -	-

Tabelle E.3.: Detaillierte Bewertung für das Studienprojekt

E.4. Internes-Projekt

	Assembla	BerliOs	Bitbucket	CodePlex	CodeSpaces	Gforge	Gitorious	Gna.org	GNU Savan	GoogleCode	GoPlan	Huddle	JavaForge	Launchpad	Origo	Ourproject	Sarovar	SourceForge	Unfuddle
Problemverfolgung	-	=	-	-	=	=	=	=	=	=	-	-	=	=	=	=	=	=	-
Dokumentenverwaltung	=	+	=	-	=	=	=	+	+	=	=	+	=	+	+	+	+	-	+
Wissenserfassung	-	+	+	+	+	+	+	=	=	+	-	+	+	+	+	+	-	+	+
Versionskontrolle	=	=	=	=	=	=	=	=	=	=	- -	-	=	=	=	=	=	=	=
Asynchron	=	=	-	=	=	=	- -	-	-	- -	=	=	=	=	=	=	=	=	=
Projektüberwachung	=	=	- -	-	=	=	-	=	=	-	-	-	=	=	=	-	-	=	-
Zugriffskontrolle	=	+	+	+	=	+	+	+	+	+	=	+	+	=	=	+	+	=	+
Projektpräsentation	=	-	- -	-	- -	=	-	-	-	-	- -	-	-	-	-	-	-	-	-
Entscheidungsfindung	-	+	-	=	-	-	-	+	+	-	-	=	-	=	-	+	+	-	=
Aufgabenverwaltung	=	=	- -	=	=	=	- -	=	=	- -	=	=	=	-	=	=	=	-	=
Zeiterfassung	+	-	-	-	+	+	-	-	-	-	+	-	+	-	-	-	-	-	-
Aufwandsschätzung	-	-	-	-	+	+	-	-	-	-	-	=	=	=	-	-	-	-	-

Tabelle E.4.: Detaillierte Bewertung für das interne Projekt

Die folgenden Tabellen listen die Ergebnisse der einzelnen Bewertungen zu Usability und Integration, der besten beiden Plattformen, der beiden Bearbeiter auf.

	Bewertung	Beschreibung
Einheitlichkeit	●	Alle Werkzeuge sehen einheitlich aus. Lediglich die Farben der Rahmen ändern sich, um dem Nutzer zu signalisieren, dass ein anderes Werkzeug verwendet wird
Lokalität	●	Alle Werkzeuge befinden sich auf der JavaForge Plattform
Durchgängigkeit	●	Zwischen den einzelne Werkzeugen lassen sich Daten austauschen bzw. verlinken
Übersicht	◐	Die Werkzeuge empfinde ich nicht optimal übersichtlich
Bedienbarkeit	◐	Die Werkzeuge lassen sich mit wenigen Klicks erreichen, sind aber nicht immer einfach und intuitiv zu bedienen
Lernbarkeit	●	Die Nutzung der Werkzeuge konnte schnell erlernt werden und ist nicht schwierig

Tabelle E.5.: Bewertung JavaForge Bearbeiter 1

	Bewertung	Beschreibung
Einheitlichkeit	●	Alle Werkzeuge sehen einheitlich aus
Lokalität	●	Alle Werkzeuge befinden sich auf der GForge Plattform
Durchgängigkeit	●	Zwischen den einzelne Werkzeugen lassen sich Daten austauschen bzw. verlinken
Übersicht	◐	Alle Werkzeuge waren direkt zu sehen und konnten in einer Menüleiste direkt ausgewählt werden. Jedoch ist es für jemanden, der die Plattform das erste mal nutzt, schwer zu seinem Projekt zu gelangen
Bedienbarkeit	●	Die Werkzeuge lassen sich mit wenigen Klicks erreichen und sind einfach und intuitiv zu bedienen. Tätigkeiten werden ebenfalls mit wenigen Klicks ausgeführt und benötigen keinen langen Prozess
Lernbarkeit	●	Die Nutzung der Werkzeuge konnte schnell erlernt werden und ist nicht schwierig

Tabelle E.6.: Bewertung GForge Bearbeiter 1

	Bewertung	Beschreibung
Einheitlichkeit	●	Alle Werkzeuge sehen einheitlich aus. Dadurch findet man sich schnell in den einzelnen Werkzeugen zu recht
Lokalität	●	Alle Werkzeuge werden von JavaForge zur Verfügung gestellt. Kein Werkzeug verlinkt auf eine externe Seite.
Durchgängigkeit	●	Die Werkzeuge können untereinander Informationen austauschen. Es ist z.B. möglich in einem Wiki-Eintrag auf einen Forum-Eintrag zu verlinken und umgekehrt
Übersicht	●	Alle Werkzeuge sind direkt zu sehen und können in einer MenÏleiste direkt ausgewählt werden
Bedienbarkeit	●	Die Werkzeuge lassen sich mit wenigen Klicks erreichen und sind einfach und intuitiv zu bedienen. Tätigkeiten werden ebenfalls mit wenigen Klicks ausgeführt und benötigen keinen langen Prozess
Lernbarkeit	●	Die Nutzung der Werkzeuge konnte schnell erlernt werden und ist nicht schwierig

Tabelle E.7.: Bewertung JavaForge Bearbeiter 2

	Bewertung	Beschreibung
Einheitlichkeit	●	Alle Werkzeuge sehen einheitlich aus
Lokalität	●	Alle Werkzeuge werden von GForge zur Verfügung gestellt und verlinken nicht auf externe Anbieter
Durchgängigkeit	●	Die Werkzeuge können untereinander Informationen austauschen. Z.B. kann in Wiki-Einträgen auf Forum-Einträge verlinkt werden
Übersicht	◐	Alle Werkzeuge sind direkt zu sehen und können in einer MenÏleiste direkt ausgewählt werden. Es bedarf ein wenig Zeit bis man sich zurecht gefunden hat.
Bedienbarkeit	●	Die Werkzeuge lassen sich einfach bedienen
Lernbarkeit	●	Die Nutzung der Werkzeuge konnte schnell erlernt werden und ist nicht schwierig

Tabelle E.8.: Bewertung GForge Bearbeiter 2

Literaturverzeichnis

[1] Karl Fogel. *Producing Open Source Software, How to Run a Successful Free Software Project*. dpunkt.verlag GmbH, 2005. (Zitiert auf den Seiten 13 und 17)

[2] Rolf Porst. *Fragebogen Ein Arbeitsbuch*. VS Verlag fr Sozialwissenschaften, 2008. (Zitiert auf Seite 28)

[3] Haggen Hau Heng So. *Construction of an Evaluation Model for Free/Open Source Project Hosting Sites*. PhD thesis, School of Business Information Technology Business Portfolio RMIT University, 2005. (Zitiert auf den Seiten 13, 17, 18 und 33)

[4] Jean Tabaka. *Collaboration Explained: Facilitation Skills for Software Project Leaders*. Addison Wesley Professional, 2006.

[5] Grady Booch und Alan W. Brown. *Advances in Computers*, volume 59, chapter Collaborative Development Environment, page 2ff. Academic Press, 2003. (Zitiert auf den Seiten 17 und 34)

[6] Filippo Lanubile und Christof Eberf und Rafael Prikladnicki und Aurora Vizcaino. Collaboration Tools for Global Software Engineering. *Software Technology*, 2010. (Zitiert auf Seite 18)

[7] James D. Herbsleb und Deependra Moitra. Global Software Development. Technical report, Lucent Technologies, 2001.

[8] Josef Altmann und Gustav Pomberger. Kooperative Softwareentwicklung: Konzepte, Modell und Werkzeuge. In *Tagungsband, 4*, 1999. (Zitiert auf den Seiten 12, 18 und 23)

[9] Jochen Ludewig und Horst Lichter. *Software Engineering*. dpunkt.verlag GmbH, 2007. (Zitiert auf den Seiten 19, 20, 23 und 24)

[10] Robert E. Kraut und Lynn A. Streeler. Coordination in Software Development. *Communications of the ACM*, 1995.

[11] Robert Haslebner und Rudolf Kussberger und Bertram Thonhause und Michael Umrutscher und Stefan Wurzinger. Software Project Hosting. Technical report, Universittszentrum Rottenmann, 2004.

[12] Austen Rainer und Stephen Gale. Sampling open source projects from portals: some preliminary investigations. *Software Metrics*, 2005. (Zitiert auf den Seiten 11, 13, 17, 19 und 20)

[13] Dieter Spath und Sven Schimpf und Andreas Kugler. Webbasierte Open-Source-Kollaborationsplattformen. Technical report, Fraunhofer Gesellschaft, 2007. (Zitiert auf den Seiten 13, 16 und 18)

[14] Michael Geisser und Tobias Hildenbrand und Lars Klimpke und Asarnusch Rashid. Werkzeuge zur kollaborativen Softwareerstellung - Stand der Technik. Technical report, Universitt Mannheim, 2006.

[15] Wikipedia. Comparison of open source software hosting facilities. http://en.wikipedia.org/wiki/Comparison_of_open_source_software_hosting_facilities, August 2005. Online, 02.04.2010. (Zitiert auf den Seiten 12 und 33)

[16] Laurie Ann Williams. Collaborative Software Process. Technical report, Department of Computer Science - The University of Utah, 2000. (Zitiert auf Seite 18)

Erklärung

Hiermit versichern wir, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Aleksandar Tolev Modood Alvi Timotheus Giuliani)