Gottfried Wilhelm
Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Praktische Informatik
Fachgebiet Software Engineering

Erfassung der Anforderungstreue in Softwareprojekten

Masterarbeit

im Studiengang Informatik

von

Sebastian Minks

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüfer: Prof. Dr. Joel Greenyer
Betreuer: Dipl.-Math. Olga Liskin

Hannover, 25.12.2013

Abstract

This thesis aims to develop a concept to assess the requirement compliance of software projects. The goal is to create a practical method that can be used with as many projects as possible. The customers of theses projects should be able to determine the requirement compliance of their projects in a time-saving and easy manner.
Before the term requirement compliance get defined the term requirement is explained.
The challenges of the topic are described and the concept gets devided into sub-concepts. During the first step the customer will record his requirements as features. Following that the customer will assign a priority to every feature. Different methods of priorisation get discussed. How do deal with changes of requirements during development get debated in the following sub-concept. Afterwards a method to evaluate the success of the entered features is constructed. The penultimate step creates a concept to deal with functions of the software that do not correspond to any of the requirements of the customer. Finally, based on the previous entered data of the other sub-concepts, a formula to compute the requirement compliance is created.
The concept gets tested with a webservice prototype and based on the results of these tests the concept gets optimized.
Following that the results are critically discussed and possible weaknesses are considered, particularly regarding misconduct in the case of edge cases as well as not-accurate customer input.

Zusammenfassung

In dieser Arbeit wird ein Konzept zur Erfassung der Anforderungstreue entwickelt. Ziel dabei ist vor allem ein praktisch anwendbares Verfahren. Kunden von Softwareprojekten sollen in der Lage sein zeitsparend und einfach die Anforderungstreue ihrer Projekte zu bestimmen. Bevor der Anforderungstreue eine Definition zugeordnet wird, wird zunächst der Begriff der Anforderung erläutert.
Anschließend werden die Herausforderungen des Themas beschrieben und das Gesamtkonzept daraufhin in Unterkonzepte aufgeteilt. Im ersten Schritt geht es darum, dass der Kunde seine Anforderungen in schriftlicher Form fixiert – dazu wird der Begriff des Feature eingeführt. Daraufhin wird erarbeitet wie der Kunde seinen Anforderungen eine Priorität zuordnen kann und was diese für Auswirkungen auf die Anforderungstreue hat. Anschließend wird ein Subkonzept entwickelt, welches die Änderung von Anforderungen während des Projektablaufs behandelt. Nach dem Projektende schließlich geht es darum den Erfolg der fixierten Anforderungen (Features) zu bewerten. Dazu werden verschiedene Möglichkeiten der Bewertung betrachtet. Im vorletzten Schritt wird erarbeitet wie mit implementierten Funktionen umgegangen wird, die keinen Anforderungen des Kunden entsprechen. Abschließend wird aufbauend auf den von den Subkonzepten gelieferten Daten der Wert für die Anforderungstreue berechnet.
Für das Konzept wird ein Webservice Prototyp entwickelt, mit welchem Tests und Kundeninterviews durchgeführt werden um das Konzept zu testen und optimieren.
Das erarbeitete Konzept wird daraufhin kritisch diskutiert und mögliche Schwachstellen werden betrachtet. Mögliches Fehlverhalten im Falle von Randfällen als auch ungenauer Kundeneingaben werden dabei besonders betrachtet.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

Wie bewertet man ob ein Projekt erfolgreich ist? Bei klassischen Projekten mit physischen Endprodukten, wie zum Beispiel einem Ventil, wird man unter anderem diverse physische Tests am gelieferten Produkt durchführen und anhand von vorher definierten Grenzwerten entscheiden, ob das gelieferte Produkt die Anforderungen erfüllt. So ist es sehr unmissverständlich möglich den Erfolg eines gegebenen Projekts zu bewerten.
Schwieriger wird es bei Projekten, die kein physisches Produkt liefern. Im Rahmen dieser Arbeit wird ein Konzept entwickelt, um den Erfolg von Softwareprojekten zu messen – doch wieso ist diese Bewertung bei Softwareprojekten so viel schwieriger? Ohne physisches Produkt kann es keine physischen Tests geben, das scheint offensichtlich. Doch selbst die Frage 'was genau sind Anforderungen?' ist schwieriger anschaulich zu beantworten, als auf den ersten Blick vermutet werden kann. Wenn es um Anforderungen geht, ist immer wesentliches Fingerspitzengefühl wichtig und in diesem Umfeld ein festes Konzept zu entwickeln birgt erhebliche Herausforderungen.
Die Softwarelandschaft entwickelt sich im Vergleich zur Herkömmlichen in einer rasenden Geschwindigkeit. Es werden nicht nur bessere Hardwarekomponenten und Programmiersprachen entwickelt, sondern auch neue Methoden zur Durchführung von Projekten, wie zum Beispiel Extreme Programming [BA04]. Um herauszufinden, wie erfolgreich ein bestimmtes Projekt (mit einer bestimmten Projektform) war, wird ein Maß für den Projekterfolg benötigt. Ein solches Maß macht es möglich sowohl die Methode selbst zu verbessern, als auch verschiedene Methoden zu vergleichen und die zukünftige Entwicklung und Forschung in eine effektive Richtung zu leiten.
Aus diesen Gründen wird in dieser Arbeit ein Konzept entwickelt die Anforderungstreue eines Projekts als solches Maß zu berechnen.

1.2 Aufgabenstellung

Folgende Ziele umfassen den Rahmen dieser Arbeit:

- Es ist ein Konzept zu entwickeln, welches die Anforderungstreue aus kleinen Teilen bestimmt.
- Es ist zu ermitteln, welchen Anforderungen ein geeignetes Werkzeug genügen muss, um das Konzept abzubilden. Aktivitäten dieser Anforderungsanalyse umfassen:

  - Interviews mit potentiellen Kunden
  - Tests an einem Webservice Prototypen
  - Untersuchungen von bisherigen Spezifikationen
- Insbesondere ist die Leichtgewichtigkeit des Konzepts und des dazugehörigen Werkzeugs wichtig. Die Durchführung und Bedienung muss schnell und mühelos möglich sein.

1.3 Aufbau der Arbeit

Zunächst werden in Kapitel 2 die Grundlagen dieser Arbeit betrachtet. Dabei werden zunächst Anforderungen von Kunden als der zentrale Punkt dieser Ausarbeitung genauer behandelt und es wird erläutert, ob Softwareprojekte im Vergleich zu herkömmlichen Projekten besonderen Umgang mit Anforderungen erfordern. Anschließend werden verschiedene Möglichkeiten betrachtet, um Anforderungen schriftlich festzuhalten.
In Kapitel 3 wird das Konzept zur Erfassung der Anforderungstreue vorgestellt und detailliert beschrieben. Für die verschiedenen Herausforderungen dieses Konzepts werden sechs eigene Subkonzepte entwickelt:

- Beschreibung von Anforderungen
- Priorisierung
- Änderungen von Anforderungen
- Bewertung
- Nicht geforderte Funktionen
- Berechnung der Anforderungstreue

Jedes dieser Subkonzepte entsteht strukturiert und wird verglichen mit verwandten Arbeiten. Anschließend wird das Konzept in seiner Gesamtheit betrachtet und anhand eines Beispiels durchgeführt.
Kapitel 4 setzt sich kritisch mit den Ergebnissen aus dem vorherigen Kapitel auseinander. Dabei werden mögliche Schwachstellen betrachtet und dessen potentielle Auswirkungen beschrieben. Weiterhin werden Alternativen zu den möglichen Kritikpunkten angeschnitten. Es wird diskutiert, ob das Konzept die Herausforderungen überwunden hat oder nicht. Außerdem wird die Rolle des Kunden in diesem Zusammenhang kritisch betrachtet.
Daraufhin wird in Kapitel 5 der entwickelte Webservice vorgestellt und beschrieben.
Zuletzt wird in Kapitel 6 ein Fazit aus den Ergebnissen dieser Arbeit gezogen. Es wird eine Zusammenfassung dieser Arbeit präsentiert und schließlich ein Ausblick auf mögliche zukünftige Forschungen gegeben.

1.4 Verwandte Arbeiten

Im Paper 'Requirements Compliance as a Measure of Project Success' [SLPK13] stellen Kurt Schneider und Olga Liskin die Idee der Anforderungstreue als Maß für den Projekterfolg vor. Sie beschäftigen sich mit Einflüssen auf den Projekterfolg und entwickeln schließlich eine Formel für die Anforderungstreue.
In Relation zu [SLPK13] stellt diese Arbeit einen stärkeren Bezug zur Praxis dar: wie kann man dieses entwickelte Maß tatsächlich praktisch auf durchgeführte Projekte anwenden? Wie gelangt man für ein beliebig gegebenes Projekt auf einen expliziten Wert der Anforderungstreue? Diese Antworten sollen im Rahmen der Arbeit beantwortet werden. Diese Arbeit damit eine Vertiefung in eine vor allem praktisch anwendbare Richtung dar.
Die Anforderungstreue hat eine enge Beziehung zum Projekterfolg. Mit diesem befasst sich [PV06]. Es werden verschiedene Faktoren betrachtet, die Auswirkungen auf den Erfolg eines Projekts haben, zum Beispiel die Einhaltung von Terminen und Finanzbudget. In dieser Arbeit werden dagegen alle diese Faktoren ignoriert und nur die Anforderungstreue betrachtet. Was diese genau bedeutet, wird im folgenden Kapitel genauer erläutert.
Da das Konzept modular in verschiedene Schritte aufgeteilt ist, sind sonstige verwandte Arbeiten direkt in die entsprechenden Unterkapitel eingefügt.

2 Grundlagen

In dieser Arbeit wird ein Konzept zur Erfassung der Anforderungstreue erarbeitet. Bevor dieser Begriff definiert werden kann stellen sich einige Fragen: was sind Anforderungen? Wer hat Kontakt mit ihnen? Wie können Anforderungen formal niedergeschrieben werden?
Diese Fragen und ihre Antworten können Auswirkungen auf die Anforderungstreue beziehungsweise auf die Berechnung dieser haben, deshalb werden sie ihm Rahmen dieser Arbeit betrachtet.

2.1 Anforderungen

Der Begriff der Anforderungen ist gefühlt sehr einfach: jeder hat eine Vorstellung davon, was damit gemeint ist und die wenigsten würden diesen Begriff wohl als missverständlich betrachten. Beschäftigt man sich allerdings intensiver damit, wird man auf eine Vielzahl verschiedener Definitionen, Anforderungen und Sonderfälle treffen.

Laut IEEE [IEEE610] ist der Begriff 'Anforderung' wie folgt definiert:
Eine Anforderung ist:

1. Eine Bedingung oder Fähigkeit, die von einem Benutzer (Person oder System) zur Lösung eines Problems oder zur Erreichung eines Ziels benötigt wird.
2. Eine Bedingung oder Fähigkeit, die ein System oder Teilsystem erfüllen oder besitzen muss, um einen Vertrag, eine Norm, eine Spezifikation oder andere, formell vorgegebene Dokumente zu erfüllen.
3. Eine dokumentierte Repräsentation einer Bedingung oder Eigenschaft gemäß (1) oder (2)

Das Thema der Anforderungen scheint so simpel und ist doch sehr komplex. Die Definition der 'Anforderung' hat selbst eine große Anzahl an Anforderungen zu erfüllen, da in der Regel sehr viele verschiedene Personen Kontakt zu Anforderungen haben und diese aus verschiedenen Blickwinkeln betrachten.
Am Fachgebiet Software Engineering der Leibniz Universität Hannover wurde das Konzept der Software Quanten entwickelt, welches diese Problem verdeutlicht, siehe Abbildung 2.1.

Der Ursprung einer Anforderung im Bereich der Softwareprojekte ist in der Regel ein Kunde, der ein Produkt haben möchte. Seine Anforderungen werden über verschiedene Stationen kommuniziert und landen letztendlich bei Entwicklern, die diese Umsetzen. Bei jedem Schritt der Kommunikation und Implementation können die ur-

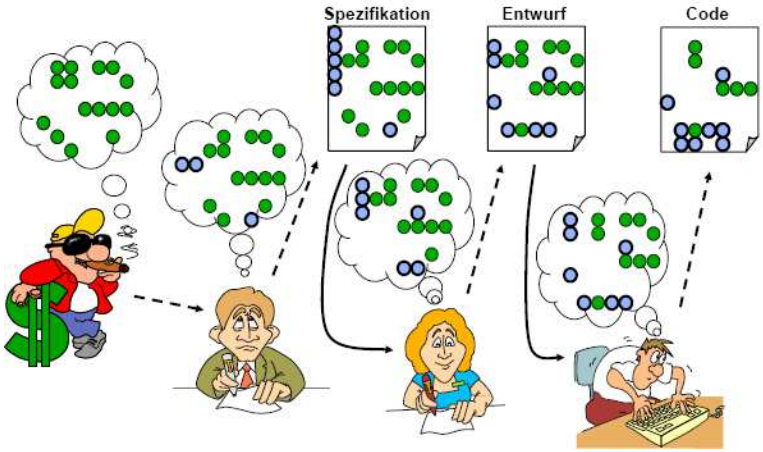

Abbildung 2.1: Software Quanten, Quelle: http://www.se.uni-hannover.de

sprünglichen Anforderungen des Kunden verändert und verfälscht werden. Das finale Softwareprodukt wird immer bis zu einem gewissen Grad von den ursprünglichen Vorstellungen des Kunden abweichen.
Im Rahmen dieser Arbeit und des entwickelten Konzepts wird der Kunde der einzige Akteur sein, deshalb möchte ich ein sehr einfachen Prozess im Bezug auf Anforderungen vorstellen:

1. Der Kunde kommuniziert seine Anforderungen nach außen.
2. Der Kunde bekommt ein Softwareprodukt geliefert.
3. Der Kunde vergleicht seine (kommunizierten) Anforderungen mit dem Produkt.

Da für diese Zwecke nur der Kunde relevant ist, können wir die IEEE Definition der Anforderung zwar benutzen, aber etwas vereinfachen und greifbarer machen:
Weniger abstrakt erklärt wird eine Anforderung ein Text sein, den der Kunde eines Softwareprojekts aufschreibt und welcher bestimmte Bedingungen an die gelieferte Software oder das Umfeld der Software hat. Später wird dieser aufgeschriebene und somit fixierte Text als Referenz gelten um zu bewerten, ob die Anforderung erfüllt ist.

2.2 Besonderheiten von Softwareprojekten

Wie im vorherigen Kapitel angeschnitten ist wird es in Bereichen, die mit Anforderungen in Berührung sind, schnell kompliziert. Gerade im Umfeld der Softwareentwicklung lassen sich Anforderungen oft nur ungenau beschreiben, da nicht wie zum Beispiel bei einem entwickelten Ventil der Wert des maximalen Durchflusses bestimmt werden kann. Solch neutrale und objektive Tests im Softwareumfeld zu machen ist äußerst schwierig, da selbst intuitiv einfach angenommene Sachverhalte extrem kompliziert sein können, siehe zum Beispiel das Halteproblem [Dav58].
Dazu kommt, dass Software beliebig komplex sein kann und Funktionen auf viele ver-

schiedene Arten implementiert werden können. Nicht nur was den Programmierstil betrifft, sondern auch den Projektablauf. Da Softwareentwicklung nicht an eine physische Entwicklungskette gebunden ist, haben Entwickler viele Freiheiten im Ablauf des Projekts. Dieser Sachverhalt hat zur Folge, dass es sehr viele Formen gibt, solche Projekte durchzuführen. Zum Beispiel klassisch nach V-Modell [Wal01] oder agil per Extreme Programming [BA04].
In diesem sich schnell entwickelnden Umfeld wäre es also besonders wichtig, neutral und objektiv bewerten zu können, welche Projekte besonders gute Ergebnisse lieferten und welche nicht. Gut bedeutet in diesem Sinne sehr nahe an den Anforderungen des Kunden, also anforderungstreu. Eine solche Bewertung erlaubt es Verbesserungen an bestehenden Methoden vorzunehmen und die Forschung in eine vielversprechende Richtung zu lenken.

2.3 Anforderungstreue

Nachdem der Begriff der Anforderung und dessen Umfeld nun klarer ist kann endlich der nächste Schritt getan werden: was ist Anforderungstreue? In [SLPK13] wird die Anforderungstreue als Idee eingeführt. Sie soll als Konzept benutzt werden, um den Erfolg von Softwareprojekten zu messen. Dabei wird sie anschaulich als eine Schnittmenge aus zwei verschiedenen Mengen betrachtet. Zum einen die Menge der Anforderungen des Kunden und zum anderen die Menge der implementierten Anforderungen des gelieferten Softwareprodukts. Dieser Sachverhalt ist in Grafik 2.2 dargestellt.

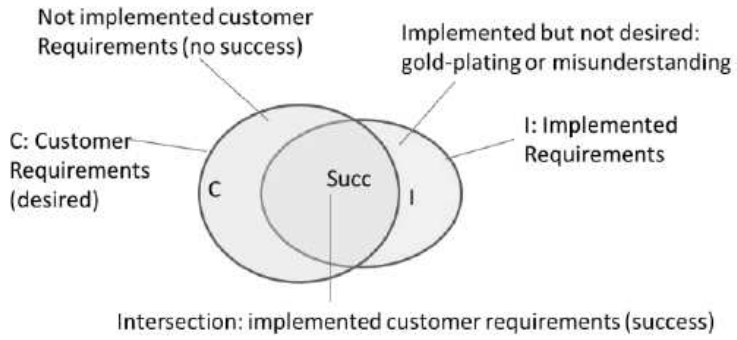

Abbildung 2.2: Anforderungstreue als Schnittmenge, Quelle: [SLPK13]

Im weiteren Verlauf von [SLPK13] wird für die Anforderungstreue folgende Formel aufgestellt:

$$RC = 1 \cdot S - 0,5 \cdot I \tag{2.1}$$

S = % der erfüllten Kundenanforderungen, I = % der nicht geforderten Anforderungen

Aufgrund dieser Formel bewegt sich der Wert für die Anforderungstreue zwischen -50 und 100.
Das Konzept, welches in Kapitel 3 dieser Arbeit entwickelt wird, wird sich deutlich von

dieser Formel und vom Definitionsbereich der Anforderungstreue entfernen, aber die extrem passende Grundidee aus Abbildung 2.2 bleibt identisch. Ebenso die Definition der Anforderungstreue: die Anforderungstreue ist ein Maß, welches die Größe vom Schnitt zwischen erfassten und implementierten Anforderungen misst.

2.4 Bisherige Verfahren

An der Leibniz Universität Hannover werden am Fachgebiet Software Engineering (Institut für Praktische Informatik) jährlich Software Projekte mit Studenten durchgeführt. Dabei werden die Studenten auf Gruppen (3-5 Personen) aufgeteilt und diese Gruppen führen anschließend ein echtes Projekt mit realen Kunden durch. Der Kunde möchte in der Regel ein Softwareprodukt zum tatsächlichen Einsatz haben. Zum Beispiel ein Webservice zur Online-Anmeldung von Klausuren, oder mobile Applikationen für diverse Zwecke. Im Rahmen dieses Projekts wird besonderes Augenmerk auf die Anforderungen der Kunden gelegt. Zunächst erarbeiten die Entwicklergruppen Spezifikationen in Rücksprache mit dem Kunden, um alle Anforderungen zu erfassen und eine schriftliche Referenz davon zu erstellen. Dieses geschieht mit Hilfe von Fließtext, als auch durch die Benutzung von definierten Methoden zur Beschreibung von Anforderungen, Abbildung 2.3[1] zum Beispiel zeigt ein Use Case Diagramm aus einem dieser Projekte.

Die Erfolgsbewertung vom Kunden findet dann in zwei Schritten statt.

1. Die schriftliche Spezifikation wird vom Kunden auf Vollständigkeit bewertet.
2. Nach Ende des Projekts werden vorher definierte Abnahmetestfälle durchgespielt und der Erfolg des Projekts auf Ergebnis dieser Tests bestimmt.

[1] Quelle: Software Projekt Complexity Navigator, WS1213 Leibniz Universität Hannover Fachbereich Software Engineering

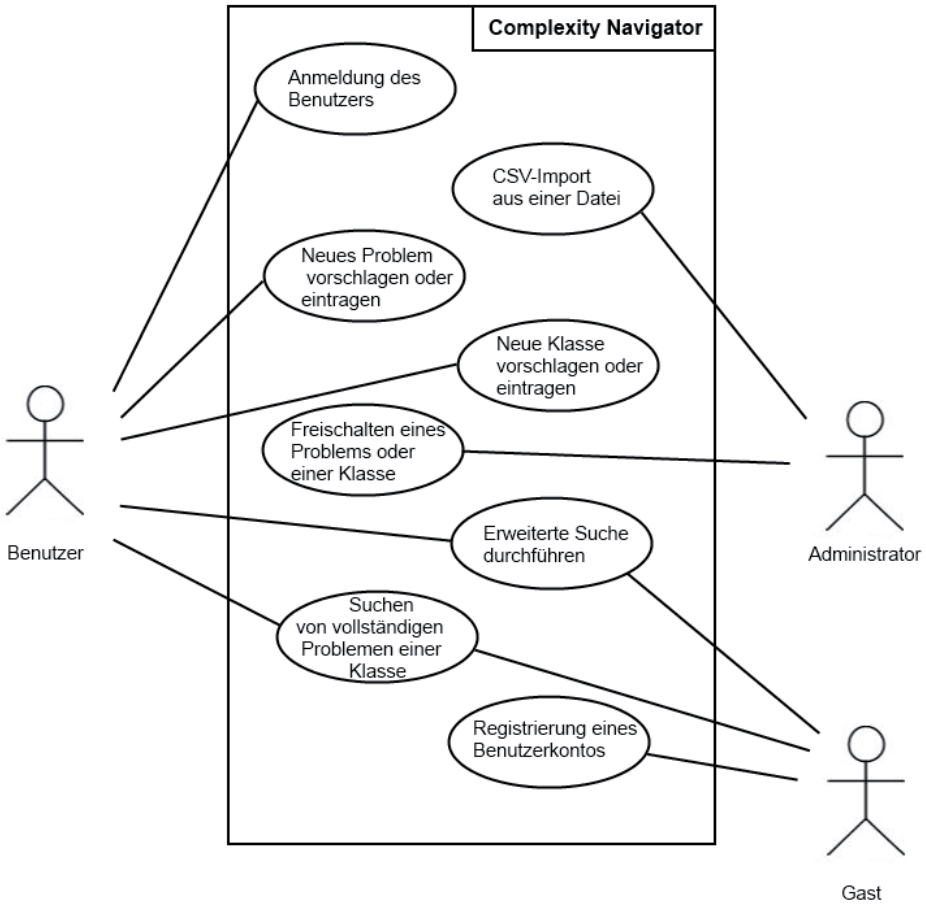

Abbildung 2.3: Use Case Diagramm aus einem Softwareprojekt

Abnahme-Testfall 1: Schnellsuche (siehe Use Case 1)
Hier möchte ein Benutzer eine Klasse mit Hilfe der Schnellsuche finden.

Eingabe	**Soll-Resultat**
Der Benutzer klickt in das Feld für die Schnellsuche	Fokussierung des Suchfeldes und Möglichkeit der Eingabe in dieses
Eingabe des Textes „Clique“	Eingabe wird in das Suchfeld übernommen
Betätigung der Eingabetaste	Eine neue Seite mit Suchergebnissen wird angezeigt. Diese sind: ○ Clique ○ Half Clique

Abnahme-Testfall 2: Eintragen eines Problems (siehe Use Case 3)
Hier möchte ein Administrator ein neues Problem in eine Klasse eintragen.

Eingabe	**Soll-Resultat**
Der Administrator wählt eine Komplexitätsklasse aus.	Die Übersicht der Probleme in dieser Klasse wird angezeigt.
Der Administrator klickt auf „Submit new problem“.	Eingabeformular für ein neues Problem wird angezeigt.
Der Administrator trägt ein: 3SAT (Name), „Formel F“ (Input), „ist die Formel F erfüllbar und in 3KNF?“ (Question), „P-M-Reduction“ (Reduction), „Logic“ (Problem category)	Warnung, dass Referenz noch fehlt, wird noch vor dem Absenden angezeigt.
Der Administrator klickt auf „Submit problem“.	Die Klassenübersicht wird zusammen mit einer Erfolgsmeldung und dem neuen Problem angezeigt.

Abnahme-Testfall 3: XML-Export (siehe Abschnitt 3.3)
Hier möchte ein Benutzer eine Ausgabe im XML-Format, um sie in anderen Programmen weiterzuverwenden.

Eingabe	**Soll-Resultat**
Der Benutzer gibt die URL ein: [domain]/np/clique.xml Dabei ist [domain] der Pfad zur Webseite, auf der die Anwendung installiert ist. np ist die ID der Komplexitätsklasse NP und clique die ID des Clique-Problems.	Alle Daten der Testdatenbank zum Problem CLIQUE werden als ein in XML-exportiertes Objekt dargestellt. Semantik: wird als Teil der Dokumentation des Programms entwickelt

Abbildung 2.4: Abnahmetestfälle aus einem Softwareprojekt

Der gesamte Prozess ist sowohl für Entwickler, als auch für den Kunden sehr zeitintensiv. Es ist zwar nicht möglich, aus diesem Verfahren direkt ein Konzept zur Erfassung der Anforderungstreue abzuleiten, aber einige Ideen können übernommen werden.
Die Spezifikation von festen Abnahmetestfällen ist wichtig und hilfreich, wenn Interaktion zwischen Entwickler und Kunde involviert ist, aber sie ist auch sehr detailliert

und zeitaufwendig. Abbildung 2.4[2] zeigt solche Testfälle. Um schnell und pragmatisch die Anforderungstreue von einem bereits durchgeführten Projekt zu erfassen, ist diese Methode daher nicht geeignet. Die Idee, Anforderungen in irgendeiner Weise schriftlich zu fixieren und dieses später als Referenz für die Bewertung zu benutzen, kann allerdings übernommen werden. Schriftliche Fixierung dient nicht nur der Referenz, sondern kann dem Kunden auch als Gedankenstütze dienen, was je nach Zeitspanne des Projekts wertvoll sein kann.
Weiterhin werden für diese Arbeit die Anforderungen an die Genauigkeit und Vollständigkeit der schriftlichen Darstellung der Anforderungen (in diesen Projekten die Spezifikation) etwas abgeschwächt werden, damit das Konzept leichtgewichtiger wird und vom Kunden ohne viel Spezialwissen angewendet werden kann. Der Fokus wird sein unkompliziert und pragmatisch die Anforderungstreue aus Sicht des Kunden zu bestimmen.

[2] Quelle: Software Projekt Complexity Navigator, WS1213 Leibniz Universität Hannover Fachbereich Software Engineering

3 Konzept

Um das Problem der Berechnung der Anforderungstreue eines Projekts zu lösen wird es, analog zum 'teile und herrsche' Vorgehen, in kleinere Unterprobleme aufgeteilt. So ist es möglich mit kleinen und einfach zu verstehenden Eingaben ein Endergebnis zu berechnen.
Das Gesamtkonzept gliedert sich in sechs Schritte, von denen der Kunde in fünf aktiv eingreifen kann und muss. Die einzelnen Schritte beziehungsweise Subkonzepte sind wie folgt definiert:

1. Fixierung der Anforderungen
   - In diesem Schritt beschreibt der Kunde die Anforderungen auf die unten erläuterte Weise und hält diese schriftlich fest.
2. Priorisierung
   - Der Kunde priorisiert die festgehaltenen Anforderungen schriftlich.
3. Änderungen von Anforderungen
   - Sollte es während des Projektablaufs zu Änderungen an den Anforderungen des Kunden kommen, so werden diese schriftlich festgehalten.
4. Bewertung
   - Nach der Implementierung bewertet der Kunde in diesem Schritt, wie zufrieden er mit der Erfüllung der einzelnen Anforderungen ist.
5. Nicht geforderte Funktionen
   - Nach Projektablauf bewertet der Kunde, ob es Funktionen im gelieferten Produkt gibt, die keiner seiner Anforderungen entsprechen. Sollte dieses der Fall sein, hat es negative Auswirkungen auf die Anforderungstreue.
6. Berechnung der Anforderungstreue
   - Im letzten Schritt werden die Ergebnisse aus den vorherigen Subkonzepten zusammengetragen und ein Wert für die Anforderungstreue wird berechnet.

Abbildung 3.1 zeigt ein UML 1.x Aktivitätsdiagramm des Konzepts.

Die Ziele des Konzepts sind im Wesentlichen einfach:

- Nach Durchführung des Konzepts soll ein möglichst realistischer Wert für die Anforderungstreue errechnet worden sein.
- Leichtgewichtigkeit

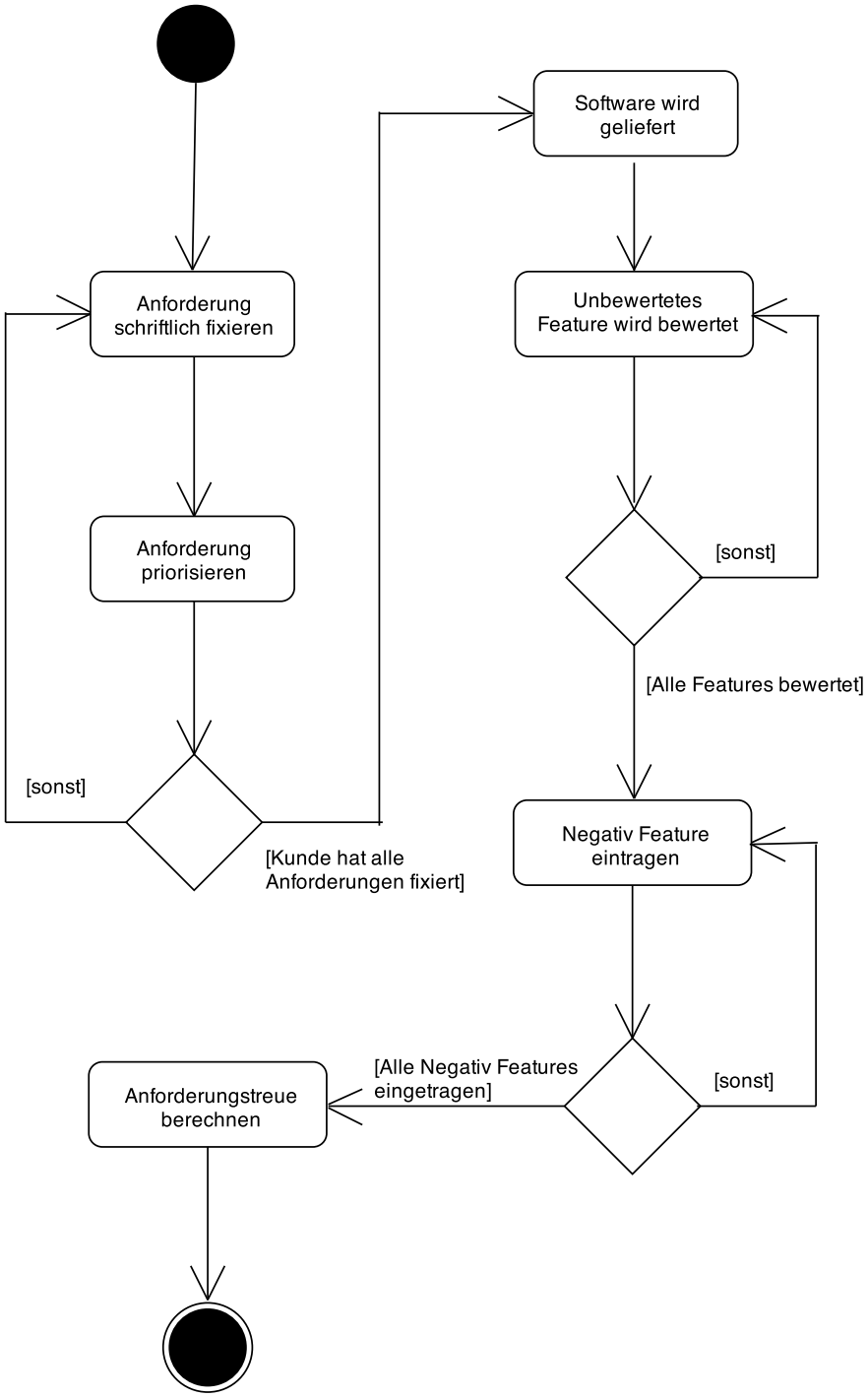

Abbildung 3.1: UML 1.x Aktivitätsdiagramm des Konzepts

– Da alle Eingaben vom Kunden kommen benötigt das Konzept Zeitaufwand vom diesem. Dieser Zeitaufwand soll möglichst gering gehalten werden.

Zunächst werden nun alle Subkonzepte beschrieben und anschließend das definierte Konzept an einem Beispielprojekt durchgeführt. Alle Subkonzepte sind dabei so aufgebaut, dass nach einer kurzen Einleitung und Erläuterung der Ziele des jeweiligen Subkonzepts erst das Verfahren vorgestellt wird, welches für diese Arbeit ausgewählt worden ist. Anschließend werden Herausforderungen und alternative Möglichkeiten betrachtet.

3.1 Subkonzepte

3.1.1 Fixierung der Anforderungen

Zunächst geht es darum die Anforderungen des Kunden festzuhalten und schriftlich zu fixieren. Dieses Subkonzept beschäftigt sich damit, in welcher Form das geschieht. Zum einen dient dieser Prozess als schriftliche Referenz am Ende des Projekts, um die gelieferte Software mit dem Fixierten zu vergleichen, zum anderen dient er als Referenz für den Kunden selbst. Dieser Schritt gliedert sich wie folgt in den Gesamtprozess ein:

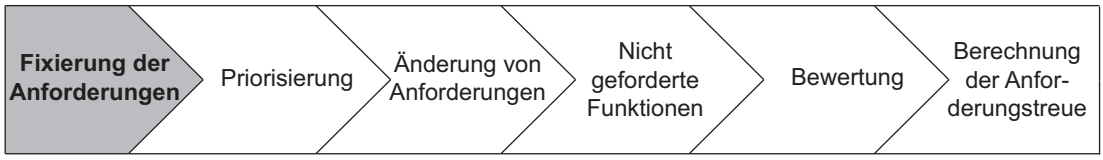

Abbildung 3.2: Eingliederung der Fixierung der Anforderungen in den Gesamtprozess

Ziele

Die Ziele dieses Subkonzepts sind:

- Leichtgewichtigkeit
  - Der Kunde soll möglichst schnell in der Lage sein seine Anforderungen zu fixieren.
- Das Konzept muss in der Lage sein beliebige Kundenanforderungen abzubilden.
  - Sowohl funktionale als auch nicht funktionale Anforderungen müssen ausdrückbar sein.
- Bewertbarkeit
  - Es soll möglichst gut vom Kunden selbst zu bewerten sein, ob eine der niedergeschriebenen Anforderungen erfüllt ist oder nicht.

Anforderungen als Features

Im Rahmen dieses Konzepts wurde entschieden Anforderungen als 'Features' zu fixieren. Um ein Gefühl dafür zu bekommen, was sich hinter diesem Begriff verbirgt seien hier einige verschiedene Definitionen erwähnt, bevor er für diese Arbeit definiert wird. Verschiedene Feature Definitionen:

- a prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems [KCHNP90]
- a distinctively identifiable functional abstraction that must be implemented, tested, delivered, and maintained [KKLKKS98]
- a distinguishable characteristic of a concept (e.g., system, component, and so on) that is relevant to some stakeholder of the concept [CE00]
- a logical unit of behaviour specified by a set of functional and non-functional requirements [Bos00]
- a product characteristic from user or customer views, which essentially consists of a cohesive set of individual requirements [CZZM05]
- a product characteristic that is used in distinguishing programs within a family of related programs [BSR04]
- an optional or incremental unit of functionality [Zav03]
- an increment of program functionality [Bat05]
- a structure that extends and modifies the structure of a given program in order to satisfy a stakeholder's requirement, to implement and encapsulate a design decision, and to offer a configuration option [ALMK08]

All diese Definitionen haben gemeinsam, dass sie Anforderungen eines Kunden mit der Funktionalität eines Programms verknüpfen. Da in diesem Schritt des Konzepts der Kunde etwas für sich selbst fixieren wird, muss eine Definition benutzt werden, die es diesem möglich macht jegliche Anforderung auszudrücken, und das ohne Einarbeitung. Es wir daher ein möglichst einfache und klare Definition verwendet:

Definition 1 (Feature) *Eine Eigenschaft oder Funktionalität eines Softwareprogramms oder dessen Umfelds (zum Beispiel Dokumentation).*

Vorteile dieser Methode sind:

- Leichtgewichtigkeit
  - Durch die sehr offene Definition kann der Kunde prinzipiell Fließtext in eigenen Worten aufschreiben.
- Alle Anforderungsarten können beschrieben werden
  - Die Definition macht es möglich funktionale, technologische, Qualitätsanforderungen, und alle sonstigen Arten von Anforderungen auszudrücken.

- Kein nötiges Spezialwissen
  - Der Kunde muss sich in keine besondere Form einarbeiten.

Allerdings ergeben sich folgende Nachteile:

- Es gibt kein hart definiertes Erfolgskriterium
  - Bei einigen unten genannten Verfahren werden explizite Erfolgsbedingungen und Tests fixiert. Während der Erfolg von implementierten Features etwas weicher im Ganzen bewertet werden muss.
- Der Begriff 'Feature' könnte dem Kunden schon bekannt sein
  - Unterschiedliche Kunden könnten verschiedene Definitionen mit dem Begriff verbinden. Dieses hat allerdings für diese Arbeit voraussichtlich wenig bis keine Auswirkung, da der Kunde selbst die Features aufschreibt und selber bewertet, also die Definition im Verlauf auf jeden Fall konsistent in den Augen des Kunden sein wird.

Wechselwirkungen mit anderen Subkonzepten

Dieser Schritt ist die Basis für alles Weitere. Ohne fixierte Anforderungen können weder Prioritäten noch Bewertungen vorgenommen werden und entsprechend kann kein Wert für die Anforderungstreue berechnet werden.

Herausforderungen und wichtige Attribute

- Leichtgewichtigkeit
  - Ein übergreifendes Ziel des gesamten Konzepts: Kundenzeit ist wertvoll und knapp, deshalb soll der nötig Einsatz möglichst gering sein.
- Kein nötiges Spezialwissen
  - Da dieser Schritt vom Kunden durchgeführt wird und dieser das Konzept ohne intensive Einarbeitung anwenden können muss, ist es nötig, dass er dieses ohne Spezialwissen durchführen kann.
- Alle Anforderungsarten müssen berücksichtigt werden können
  - Ob funktionale, technologische, Qualitätsanforderungen, oder alle sonstigen Arten von Anforderungen, sie müssen durch dieses Subkonzept beschreibbar sein.
- Bewertbarkeit
  - Es muss für den Kunden möglich sein nach Projektablauf zu bewerten, ob die fixierten Anforderungen erfüllt sind oder nicht.

Vergleich verschiedener Verfahren / Verwandte Arbeiten

Folgende alternative Methoden zur Beschreibung und Fixierung von Anforderungen wurden betrachtet:

Story Cards

Story Cards [Coh04] werden intensiv in agilen Projektformen wie zum Beispiel Extreme Programming [BA04] verwendet. Eine Story Card besteht dabei in der Regel aus zwei Elementen: einem Text, der User Story, und einer Priorität (abhängig von der jeweiligen Projektform, deshalb hier erst mal nicht relevant). Die User Story ist ein Text in einer festen Form: 'Als <Rolle> möchte ich <Ziel/Wunsch>, um <Nutzen>' [Coh04]. Normalerweise sind User Storys kurz, sodass ein Projekt aus sehr vielen Story Cards bestehen kann.
Bei der Benutzung dieser Methode entstehen folgende Vorteile:

- Alle Anforderungsarten können beschrieben werden
  - Die Definition macht es möglich funktionale, technologische, Qualitätsanforderungen, und alle sonstigen Arten von Anforderungen auszudrücken.
- Kein nötiges Spezialwissen
  - Der Kunde muss sich in keine besonders schwierige Form einarbeiten.

Story Cards bedeuten im Rahmen dieser Arbeit folgende Nachteile:

- Es gibt kein hart definiertes Erfolgskriterium
  - Bei einigen unten genannten Verfahren werden explizite Erfolgsbedingungen und Tests fixiert. Während der Erfolg von implementierten Story Cards etwas weicher im Ganzen bewertet werden muss.
- Eine sehr hohe nötige Anzahl von Story Cards ist nötig
  - Die reine Anzahl an nötigen Story Cards für ein Projekt könnte es zeitintensiv und gegebenenfalls unübersichtlich für den Kunden machen.

Use Case Beschreibungen

Eine Use Case (Anwendungsfall) Beschreibung [Coc03] ist eine Liste von Schritten, welche Interaktionen zwischen bestimmten Nutzerrollen und einem System beschreiben. In der Regel steht am Ende eines Use Cases ein explizites Ziel, welches durch die festgehaltenen Schritte erreicht werden soll. Abbildung 3.3 [1] zeigt ein Beispiel, welches im Rahmen eines Softwareprojekts des Fachbereiches Software Engineering an der Leibniz Universität Hannover entstanden ist.

Use Case Beschreibungen sind üblicherweise sehr detailliert und schildern einen Ablauf lückenlos. Es ist ebenfalls möglich ein explizites Erfolgskriterium festzuhalten.
Folgende Vorteile bieten sich bei dieser Methode:

[1] Quelle: Software Projekt Complexity Navigator, WS1213 Leibniz Universität Hannover Fachbereich Software Engineering

Use Case ID	**01**
Titel	Anmelden
Status	Entwurf
Systemgrenzen (Scope)	HTML-Seite und Menüführung
Vorbedingung	Server ist verfügbar
Mindestgarantie	
Erfolgsgarantie	Admin meldet sich erfolgreich an.
Stakeholder u. Interessen	Admin möchte die Datenbank verwalten
Hauptakteur	Admin will sich im System anmelden und die Datenbank verwalten.
Auslöser	Admin gibt die Zugangsdaten ein und klickt auf login-Button.
Hauptszenario	1. Admin ruft die Web-Anwendung auf. 2. Admin drückt auf Adminstrator-login-Button. 3. Anmeldeseite für Administrator wird angezeigt. 4. Admin gibt Benutzername in das dafür vorgesehene Feld ein. 5. Admin gibt Passwort in das dafür vorgesehene Feld ein. 6. Admin klickt auf Login-Button. 7. Die Administrator-Startseite wird angezeigt.
Erweiterungen	7a. WENN der Benutzername oder das Passwort falsch sind, DANN zurück zu Schritt 3 mit der Meldung „Login incorrect“.
Priorität	Wichtig
Verwendungshäufigkeit	Regelmäßig

Abbildung 3.3: Use Case Beschreibung

- Das Erfolgskriterium ist sehr leicht bewertbar
  - Da bei Use Cases solche Kriterien angegeben werden ist es sehr einfach, den Erfolg zu bewertet. Entweder ist das Kriterium erfüllt, oder eben nicht. Dieses Verfahren ist weniger schwammig, als die oben vorgestellten.
- Wenig nötiges Spezialwissen
  - Der Kunde muss zwar bei der Eingabe eine gewisse Form einhalten, wenn dieses allerdings über ein festes Formular per Werkzeug geschieht, dann ist die Wissenshürde weiterhin sehr gering.

Allerdings ergeben sich folgende Nachteile:

- Detaillierte Eingabe nötig
  - Da die Form der Use Case Beschreibungen sehr viele Detaileingaben voraussetzt, muss der Kunde voraussichtlich viele Daten eingeben.

- Zeitintensiv
  - Im Vergleich zu den oben genannten Verfahren ist es sehr zeitintensiv, Use Case Beschreibungen zu erstellen.
- Unklar, wie mit nicht funktionale Anforderungen dargestellt werden können
  - Da in der Regel Interaktionen mit Use Cases beschrieben werden, ist es unklar, wie nicht-funktionale Anforderungen mit diesem Verfahren ausgedrückt werden können.

Begründung der Auswahl

In Tabelle 3.1 ist eine Übersicht über die vorgestellten Verfahren im Bezug auf die Ziele des Subkonzepts gegeben.

	Feature	Story Card	Use Case Beschreibung
Leichtgewichtigkeit	✓	O	–
Bewertbarkeit	O	O	✓
Alle Anforderungsarten möglich	✓	✓	–

Tabelle 3.1: Vergleich der Beschreibungsmethoden

Leichtgewichtigkeit: wenn der Kunde Anforderungen als Feature beschreibt, kann er schreiben was und wie viel er möchte. Theoretisch können beliebig viele (ob nun sinnvoll oder nicht) Funktionen in einem Feature zusammengefasst werden. Dieses Verfahren ist deshalb so leichtgewichtig wie es nur geht, da der Kunde sich nicht einarbeiten muss und in seiner eigenen Sprache Texte verfassen kann. Story Cards haben hier im Vergleich einen Nachteil. Eine Story Card für sich ist prinzipiell zwar ebenfalls sehr einfach und leichtgewichtig, aber es ist die schiere Anzahl an nötigen Story Cards, die bei diesem Verfahren einen Nachteil bedeuten. Je mehr Story Cards angelegt werden müssen, desto mehr müssen später priorisiert und bewertet werden, was immer weiteren Zeitaufwand nach sich zieht. Use Case Beschreibungen sind sehr zeitintensiv zu erstellen, da sie sehr viele und genaue Eingaben erfordern.
Bewertbarkeit: Features sind, da sehr frei gestaltet, bis zu einem gewissen Grad schwammig und können beliebig viele Einzelanforderungen und Funktionen beinhalten. Davon könnten zum Beispiel einige erfüllt sein und andere nicht. Es ist zwar möglich subjektiv zu sagen, ob man mit einem Feature zufrieden ist oder nicht, aber ein genaues Kriterium gibt es nicht. Es ist also eine gewisse Intuition oder ein gewisses Fingerspitzengefühl vom Benutzer des Konzepts nötig. Für Story Cards zeigt sich ein ähnliches Bild. Eine gewisse Ungenauigkeit bleibt, aber durch die – im Vergleich zu Features – kurze, abgespeckte Form etwas übersichtlicher, jedoch bestehen die selben prinzipiellen Probleme auch hier. Use Case Beschreibungen sind durch ihre eingeschränkte Form und durch ihre potentiell existierende Erfolgsbedingung sehr einfach zu bewerten. In der Regel geht man in der gelieferten Software die Schritte der Use Case Beschreibung nach und muss genau zu dem fixierten Ergebnis kommen, sonst

ist diese Anforderung nicht erfüllt.
Möglichkeit verschiedene Anforderungsarten auszudrücken: die Feature Definition bietet enorme Freiheiten, so dass man damit beliebige Anforderungen ausdrücken kann. Story Cards sind ebenfalls sehr frei und können quasi alles ausdrücken, allerdings ist die Form ein wenig einschränkender als bei Features. Use Case Beschreibungen schließlich haben Probleme mit nicht funktionalen Anforderungen, da diese zum Teil nicht durch Interaktionen in der Schritt für Schritt Weise der Use Case Beschreibungen ausdrückbar sind.

Für dieses Konzept habe ich als Verfahren für die Beschreibung der Anforderungen den Begriff der Features ausgewählt. Zum einen erfüllt dieses Verfahren am besten von allen Alternativen das wichtige Ziel der Leichtgewichtigkeit und zum anderen bietet es eine enorme Flexibilität. Diese Flexibilität wird wichtig und nötig sein, da damit zu rechnen ist, dass zwischen den verschiedenen Kunden, die dieses Konzept benutzen werden, große Unterschiede in allen Bezügen existieren werden.

3.1.2 Priorisierung

Die Priorität einer Anforderung im Rahmen dieser Arbeit bedeutet die Wichtigkeit der Anforderung in den Augen des Kunden. Die Dringlichkeit spielt dabei keine Rolle, da die Anforderungstreue nur zu bestimmten Zeitpunkten (zum Beispiel nach Ende eines Projekts) gemessen wird und es deshalb keinen Unterschied macht, wann in diesem Zeitraum eine Anforderung erfüllt worden ist.
Im Rahmen dieses Subkonzepts werden Features priorisiert, um die vom Kunden vergebene Gewichtung in die Berechnung der Anforderungstreue einzubeziehen. Dieser Schritt gliedert sich wie folgt in den Gesamtablauf ein:

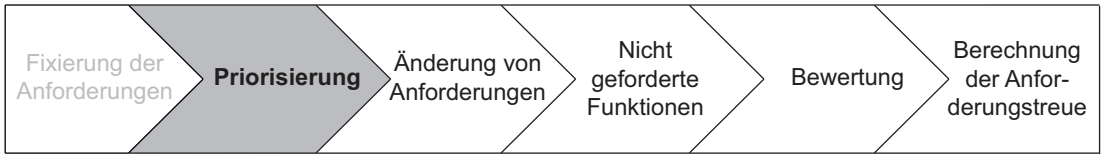

Abbildung 3.4: Eingliederung der Priorisierung in den Gesamtprozess

Ziele

Die Ziele dieses Subkonzeptes sind schnell erklärt: Die Priorität der Anforderungen in den Augen des Kunden soll möglichst genau festgehalten werden. Und zwar möglichst wenig nötigem Zeitaufwand vom Kunden selbst.

MoSCoW Methode

Die Priorisierungsmethode, die in diesem Subkonzept benutzt wird, ist im Wesentlichen eine Variante der MoSCoW Methode [CB04]. Mit diesem Verfahren können Stakeholder die Wichtigkeit von Anforderungen bestimmen. Dazu werden die Anforderungen einer von vier möglichen Prioritätsstufen zugeordnet:

- MUST
  - Diese Stufe beschreibt Anforderungen die unter allen Umständen erfüllt werden müssen und kritisch für den Erfolg des Projekts sind. Sobald eine MUST Anforderung nicht erfüllt ist sollte das Projekt als Fehlschlag gewertet werden.
- SHOULD
  - Anforderungen dieser Stufe haben ebenfalls eine hohe Priorität, sollten allerdings erfüllt werden, wenn bereits alle MUST Anforderungen erfüllt worden sind.
- COULD
  - Diese Stufe repräsentiert Anforderungen die wünschenswert aber nicht notwendig sind, sie werden erfüllt wenn MUST und SHOULD Anforderungen bereits erfüllt sind und noch Ressourcen übrig sind.
- WONT
  - WONT Anforderungen werden nicht implementiert, aber potentiell für die Zukunft berücksichtigt

Die WONT Priorität wird im Rahmen der Bestimmung der Anforderungstreue nicht benötigt, da wir nur Anforderungen bewerten und messen können, die implementiert werden sollten. Der Kunde hat also die Möglichkeit die angelegten Features in die Stufen MUST, SHOULD und COULD einzuordnen.
Diese sehr einfache und limitierte Art der Priorisierung macht es dem Kunden sehr leicht, Prioritäten voneinander abzugrenzen und Zeit zu sparen.

Insgesamt bieten sich bei diesem Verfahren folgende Vorteile:

- Leichte Abgrenzung verschiedener Prioritäten
- Niedriger Zeitaufwand für den Kunden, da die Möglichkeiten begrenzt und intuitiv sind
- Robust gegen Fehler, da kein spezielles Wissen vom Kunden gefordert wird
- Den Prioritätsstufen können relative Faktoren für die spätere Berechnung der Anforderungstreue zugeordnet werden

Dagegen stehen folgende Nachteile:

- Keine feine Gewichtung möglich, drei Stufen sind sehr grob

- Über die Zuordnung der Prioritätsstufen zu Faktoren für die spätere Berechnung der Anforderungstreue aus den gewählten Prioritäten hat der Kunde keine Kontrolle

Wechselwirkungen mit anderen Subkonzepten

Die Priorisierung hat direkte Wechselwirkungen mit den Subkonzepten Beschreibung von Anforderungen und Berechnung der Anforderungstreue. Ohne eingegebene Features im ersten Schritt können keine Prioritäten zugewiesen werden. Für die Berechnung der Anforderungstreue werden die Prioritäten durch Faktoren ersetzt. Wie das erfolgt wird in 'Berechnung der Anforderungstreue' beschrieben.

Herausforderungen und wichtige Attribute

Aus dem Vorhergegangenen ergeben sich folgende Herausforderungen und wichtige Attribute dieses Konzepts:

- Leichtgewichtigkeit
  - Ein übergreifendes Ziel des gesamten Konzepts: Kundenzeit ist wertvoll und knapp, deshalb soll der nötig Einsatz möglichst gering sein
- Berechenbarkeit in 'Berechnung der Anforderungstreue'
  - Da die Priorisierung direkt in die späteren Formeln zur Berechnung der Anforderungstreue einfließt ist diese Eigenschaft absolut notwendig
- Kein nötiges Spezialwissen
  - Da die Priorisierung vom Kunden durchgeführt wird und dieser das Konzept ohne intensive Einarbeitung anwenden können muss, ist es nötig, dass er die Priorisierung ohne Spezialwissen durchführen kann

Vergleich verschiedener Verfahren und verwandte Arbeiten

Folgende Methoden zur Priorisierung wurden in dieser Arbeit betrachtet:

Gewichtung per Faktor
Bei dieser Methode wird für jedes Feature ein Faktor zwischen zum Beispiel 0% und 100% vergeben. Durch das Verhältnis zwischen den Werten der einzelnen Features und der Summe der insgesamt vergebenen Features ergibt sich die Abstufung der Priorität zwischen den Features. Anhand dieser relativen Prioritätsverteilung kann dann ein Faktor für die spätere Gesamtberechnung errechnet werden. Diese Methode bietet folgende Vorteile:

- Eine sehr feine Abgrenzung der Priorität ist möglich

- Der Faktor für die spätere Berechnung der Priorität in der Formel für die Anforderungstreue ergibt sich aus der relativen Gewichtung, die sich aus den vergebenen Prioritätsfaktoren errechnet

Dagegen stehen folgende Nachteile:

- Eine Abgrenzung zwischen verschiedenen Prioritäten ist sehr schwierig - die Entscheidung ob eine Priorität zum Beispiel 70% oder 71% ist, ist nahezu unmöglich
- Die sehr feinen Auswahlmöglichkeiten bedeuten eine höhere Zeitinvestition für den Kunden
- Diese Methode ist sehr abstrakt und wenig anschaulich, daher steigt das Risiko für falsche Eingaben

Paarweiser Vergleich
Bei dieser Methode [Hei83] priorisiert der Kunde die Anforderungen, indem er sie paarweise vergleicht. Dabei bestimmt er für jedes Paar die wichtigere Anforderung. Diese Vergleiche werden so oft wiederholt, bis alle paarweise verschiedenen Anforderungen verglichen worden sind. Das Ergebnis ist eine Reihenfolge der Anforderungen in Bezug auf ihre relative Priorität zueinander.

Für dieses Verfahren ergeben sich folgende Vorteile:

- Der Kunde muss keine Priorität aktiv einer bestimmten Stufe oder einem bestimmten Faktor zuordnen, sondern kann durch viele kleine intuitive Entscheidungen Prioritäten vergeben
- Durch die resultierende Reihenfolge entstehen quasi so viele Prioritätsstufen wie es bewertete Anforderungen gibt, dadurch ist eine sehr feine Abgrenzung der verschiedenen Prioritäten möglich

Diesen Vorteilen stehen allerdings einige Nachteile entgegen:

- Es ist nicht möglich gleich wichtige Anforderungen zu bestimmen
- Je mehr Anforderungen existieren, desto mehr Vergleiche müssen durchgeführt werden, dadurch müssen mehr als eine linare Anzahl von Aktionen im Bezug auf die Anzahl der Anforderungen vom Kunden durchgeführt werden
- Die resultierende relative Priorität ist eine Reihenfolge ohne genaue Faktoren, diese müssen vom Konzept oder Werkzeug vorgegeben werden, ohne dass der Kunde darauf Einfluss hat

Relativer Vergleich
Diese Möglichkeit ist eine Verbindung des paarweisen Vergleichs und der MoSCoW Methode. Der Kunde sortiert die Features in abstrakte Stufen ein und hat dabei maximal so viele Stufen wie es Anforderungen gibt. Sind mehrere Features gleich wichtig

werden sie in die gleiche Stufe einsortiert. Durch die Einsortierungen ergibt sich eine relative Priorisierung der Features. Abbildung 3.5 stellt dieses Verfahren an einem Beispiel dar.

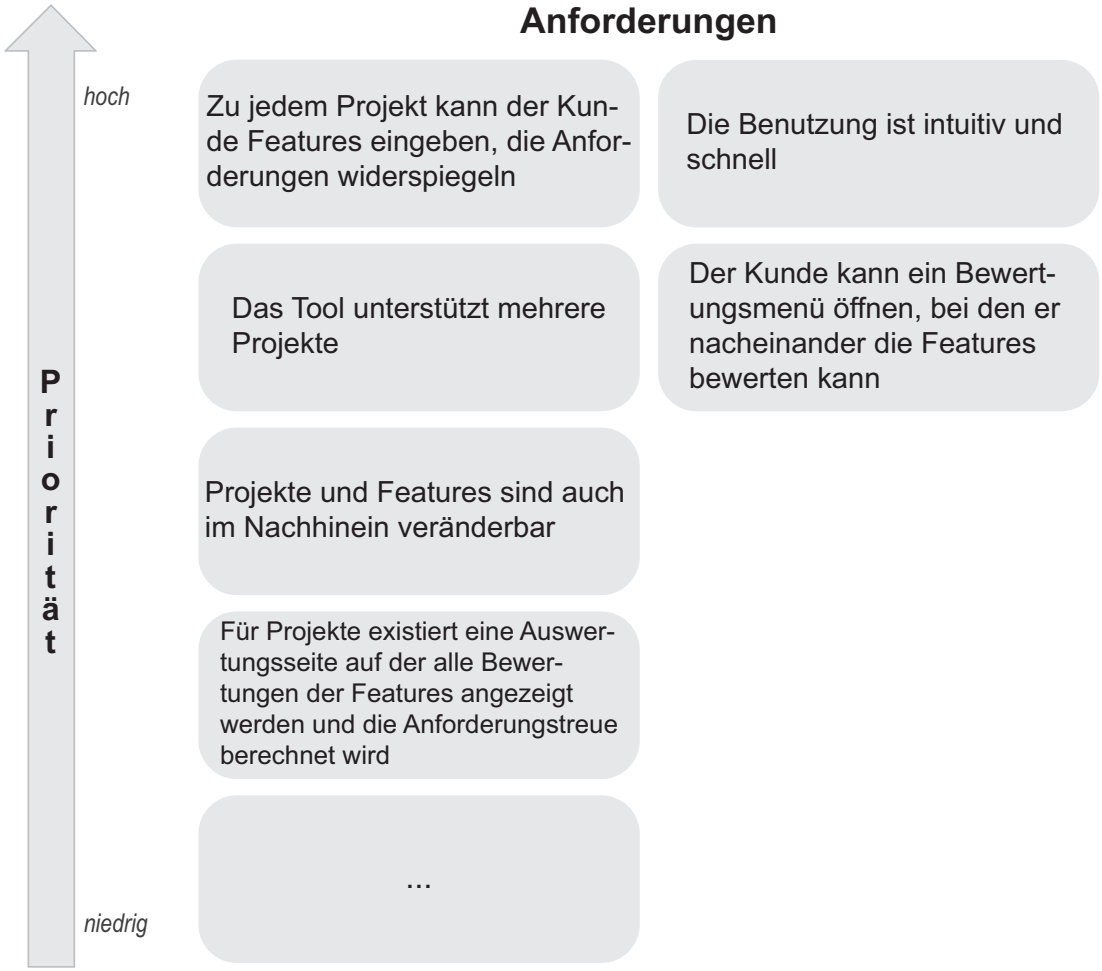

Abbildung 3.5: Beispiel für relativen Vergleich

Die Vorteile dieses Verfahrens sind:

- Der Kunde muss keine Priorität aktiv einer bestimmten Stufe oder einem bestimmten Faktor zuordnen, sondern kann durch viele kleine intuitive Entscheidungen Prioritäten vergeben
- Durch die resultierende Reihenfolge entstehen quasi so viele Prioritätsstufen wie es bewertete Anforderungen gibt, dadurch ist eine sehr feine Abgrenzung der verschiedenen Prioritäten möglich

Dem entgegen stehen folgende Nachteile:

- Bei dieser Methode muss der Kunde das Problem der Prioritätsvergabe aller Anforderungen in einer komplexen Aktion ausführen, er hat nicht die Möglichkeit

Schritt für Schritt vorzugehen und Teile des Problems auszublenden

- Die resultierende relative Priorität ist eine Reihenfolge ohne genaue Faktoren. Diese müssen vom Konzept oder Werkzeug vorgegeben werden, ohne dass der Kunde darauf Einfluss hat

Begründung der Auswahl

In Tabelle 3.2 ist eine Übersicht über die vorgestellten Verfahren im Bezug auf die Ziele des Subkonzepts gegeben.

	MoSCoW	Faktor	Paarweise	Relativ
Leichtgewichtigkeit	✓	O	–	–
Berechenbarkeit	O	✓	O	O
Kein nötiges Spezialwissen	✓	✓	✓	✓

Tabelle 3.2: Vergleich der Priorisierungsmethoden

Leichtgewichtigkeit: das MoSCoW Verfahren bietet enorme Zeitersparnis, da es zum einen sehr wenige Auswahlmöglichkeiten für den Kunden gibt und zum anderen die Prioritätsstufen sehr intuitiv sind. Egal auf welchem Kenntnisstand der Kunde ist wird er bewerten können, was für ihn zum Beispiel ein Must-have oder ein Could-have ist. Gewichtung per Faktor hat einen ähnlichen Ablauf wie MoSCoW, indem jedem Feature direkt eine Priorität gegeben wird, allerdings benötigt dieses Verfahren durch die extreme Feinheit (zum Beispiel 0% - 100%) der möglichen Werte mehr Nachdenken und Zeit vom Kunden. Der paarweise Vergleich ist zeitintensiv, da für jede angelegte Anforderung mehrere Vergleiche durchgeführt werden müssen. Anschaulich betrachtet muss der Kunde für jede Anforderung, die er anlegt, mehr als eine Aktion tätigen, um diese Anforderung zu priorisieren. Ein ähnliches Bild zeigt sich beim relativen Vergleich: auch hier muss der Kunde für jede angelegte Anforderung mehrere Vergleiche durchführen, auch wenn er diese nicht Schritt für Schritt einzeln macht, sondern – je nach Ausgestaltung des benutzten Werkzeugs – gegebenenfalls gleichzeitig.
Berechenbarkeit: denken wir einige Schritte voraus, muss es möglich sein die Priorität möglichst sinnvoll in die Formel für die Anforderungstreue einfließen zu lassen. Bei der MoSCoW Methode muss das Konzept den einzelnen Prioritätsstufen Werte für die Priorität vorgeben auf die der Kunde keinen Einfluss hat. Die Stufe MUST könnte zum Beispiel 10 fiktive Punkte wert sein, während COULD nur 1 Punkt wert sein könnte. Durch diesen Einfluss seitens des Konzepts könnte es zu einer Beeinflussung des Endergebnisses kommen, wenn der Kunde andere Relationen zwischen den Stufen annimmt. Priorisierung per Faktor hat in dieser Hinsicht einen großen Vorteil, da die vergebenen Werte vom Kunden direkt als relative Prioritätswerte der Anforderungen in der späteren Formel benutzt werden können. Deshalb ist seitens des Konzepts kein Einfluss nötig und die Prioritätsfaktoren in der Formel kommen direkt vom Kunden. Der paarweise und relative Vergleich haben das selbe prinzipielle Problem wie die MoSCoW Methode: das Konzept muss bis zu einem gewissen Grad vorgeben, wie welche

Stufe der Priorität in die spätere Formel übertragen wird.
Kein nötiges Spezialwissen: Keines der Verfahren benötigt Spezialwissen, allerdings bietet insbesondere die MoSCoW Methode eine sehr intuitive Bedienung, da sie nicht mit abstrakten Werten arbeitet sondern mit sprechenden Bezeichnungen.

Ich habe mich dafür entschieden, die MoSCoW Methode zu verwenden. Wie schon im vorherigen Subkonzept ist der wesentliche Grund die Leichtgewichtigkeit. Der Nachteil, dass die drei Prioritätsstufen vom Konzept selbst Faktoren zugeordnet werden müssen, ist ein Kompromiss, der bewusst zugunsten der Leichtgewichtigkeit eingegangen wird.

3.1.3 Änderung von Anforderungen

Dieses Subkonzept befasst sich mit der Situation, dass schon fixierte Anforderungen sich während des Projektablaufs ändern. In Abbildung 3.6 ist dieser Schritt in den Gesamtablauf eingegliedert.

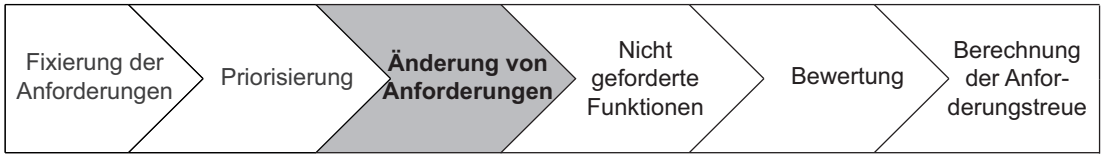

Abbildung 3.6: Eingliederung der Änderung von Anforderungen in den Gesamtprozess

Ziele

Die Ziele dieses Subkonzepts sind zum einen, dass der Kunde überhaupt in der Lage ist, angelegte Anforderungen zu ändern und zum anderen, dass die Änderungen in irgendeiner Weise in die spätere Berechnung der Anforderungstreue einfließen können.

Alleinig die neuste Version einer Anforderung beachten

Diese Methode, die in diesem Konzept benutzt wird, verwirft kompromisslos alle alten Versionen und gibt an die späteren Subkonzepte, insbesondere an die Berechnung der Anforderungstreue, nur die neuste Version einer Anforderung weiter. Für die Anforderungstreue ist mit diesem Verfahren also nur relevant welche Anforderungen zum Projektende tatsächlich existieren, und nicht welche eventuell zu anderen Zeitpunkten existiert haben.
Vorteile dieser Methode:

- Leichtgewichtigkeit
  - Der Kunde muss praktisch nichts weiter tun als den Text einer Anforderung

zu ändern, der Rest wird vom Konzept übernommen. Dadurch wird der nötige Zeiteinsatz minimiert.

- Sehr einfach in späterer Formel zu berechnen
  - Durch die Verwerfung aller alten Versionen von Anforderungen ist dieses Verfahren sehr leicht im letzten Schritt umzusetzen, um einen Wert für die Anforderungstreue zu berechnen.
- Es werden nur tatsächliche Anforderungen am Ende des Projekts betrachtet
  - Durch diesen Sachverhalt ergibt sich eine tatsächliche Momentaufnahme für die Anforderungstreue.

Nachteile dieser Methode:

- Dieses Verfahren könnte subjektiv als sehr unfair empfunden werden
  - Stellen wir uns das Szenario vor, dass der Kunde einen Tag vor Projektende eine wesentliche Anforderung massiv ändert, die bis dahin erfolgreich erfüllt war. In so einer Situation bliebe den Entwicklern natürlich nicht mehr genug Zeit, die neue Anforderung erfolgreich zu implementieren.

Wechselwirkungen mit anderen Subkonzepten

Die Priorisierung hat direkten Einfluss auf alle anderen Subkonzepte. Sowohl Beschreibung als auch Priorität und Bewertung können verändert werden. Je nachdem wie dieses Subkonzept mit Änderungen umgeht, gibt es auch Auswirkungen für die spätere Berechnung der Anforderungstreue.

Herausforderungen und wichtige Attribute

Folgende Herausforderungen muss dieses Konzept erfüllen:

- Leichtgewichtigkeit
  - Alle Folgen von Änderungen an Anforderungen müssen für den Kunden mit möglichst wenig nötigen Eingaben verbunden sein. Im Idealfall ist das Verhalten dieses Subkonzepts für den Kunden unsichtbar.
- In späterer Folge berechenbar
  - Das Subkonzept muss in der späteren Formel der Berechnung der Anforderungstreue abzubilden sein.

Vergleich verschiedener Verfahren

Folgende alternative Möglichkeiten wurden während dieser Ausarbeitung betrachtet:

Alle Versionen einer Anforderung berücksichtigen
Bei diesem Verfahren werden alle alten Versionen einer Anforderung gespeichert und als komplett eigenständige und gleichberechtigte Anforderungen betrachtet. Durch dieses Verhalten wird es zum gewissen Grad gewürdigt, wenn Anforderungen geliefert werden, die mal existiert haben aber mittlerweile verändert worden sind. Vorteile dieser Methode:

- Einfach in späterer Formel zu berechnen
  - Da alle Versionen unabhängig bewertet werden ist dieses Verfahren ebenfalls problemlos für die Berechnung eines Wertes für die Anforderungstreue einsetzbar.
- Würdigung von Entwicklungszeit
  - Es werden Entwicklungen gewürdigt für Anforderungen, die tatsächlich existiert haben und während des Projektablaufs verändert worden sind.

Nachteile dieser Methode:

- Unnötiger Zeitaufwand
  - Der Kunde muss Anforderungen auf Erfolg bewerten, die veraltet sind und unter Umständen für ihn gar nicht mehr relevant sind. Das stellt für ihn einen unnötigen Zeitaufwand dar.

Zeitpunkt der Änderung als Gewichtsmaß für Versionen von Anforderungen
Diese Methode verfährt ähnlich wie das eben vorgestellte Verfahren. Veraltete Versionen einer Änderung werden gespeichert und am Ende eines Projekts auch vom Kunden auf Erfolg bewertet. Dabei berücksichtigt wird allerdings der Zeitpunkt der Änderung. Dieser in Relation zum Projektanfang und Projektende dient als Gewichtsmaß für die spätere Berechnung, welche Anforderung wie stark in die Formel eingeht. Sollte zum Beispiel eine Änderung einer Anforderung zu einem Zeitpunkt nach zeitlich 70% des Projekts stattfinden, so geht die alte Version der Anforderung zu 70% und die neue Version lediglich zu 30%. Die Vor- und Nachteile sind ähnlich zur vorherigen Methode. Vorteile:

- Einfach in späterer Formel zu berechnen
  - Da alle Versionen unabhängig bewertet werden und lediglich der relative Zeitpunkt als Faktor hinzukommt ist dieses Verfahren ebenfalls problemlos für die Berechnung eines Wertes für die Anforderungstreue einsetzbar.
- Würdigung von Entwicklungszeit
  - Es werden Entwicklungen gewürdigt für Anforderungen, die tatsächlich existiert haben und während des Projektablaufs verändert worden sind.

Nachteile:

- Unnötiger Zeitaufwand
  - Der Kunde muss Anforderungen auf Erfolg bewerten, die veraltet sind und

unter Umständen für ihn gar nicht mehr relevant sind. Das stellt für ihn einen unnötigen Zeitaufwand dar.

Begründung der Auswahl

In Tabelle 3.3 ist eine Übersicht über die vorgestellten Verfahren im Bezug auf die Ziele des Subkonzepts dargestellt.

	Alte verwerfen	Alle gleichberechtigt	Zeit als Gewichtsmaß
Leichtgewichtigkeit	✓	O	O
Berechenbarkeit	✓	✓	✓

Tabelle 3.3: Vergleich der Methoden zum Umgang mit Anforderungsänderungen

Leichtgewichtigkeit: Das Verfahren alle alten Versionen von Anforderungen nach einer Änderung zu verwerfen ist mit Sicherheit die leichtgewichtigste. Die einzige Aktion, die der Kunde durchführen muss, ist das ändern der Anforderung. Beide anderen Verfahren benötigen zusätzliche Aktionen, da der Kunde alle alten Anforderungsversionen zusätzlich auf Erfolg bewerten muss.
Berechenbarkeit: Alle drei verfahren lassen sich für eine spätere Berechnung problemlos in eine mathematische Formel übertragen.

Im Rahmen dieser Arbeit wurde das Verfahren zum Umgang mit Anforderungsänderungen ausgewählt, welches alle alten Versionen einer Anforderung verwirft und zur Berechnung der Anforderungstreue alleinig die neueste Version dieser benutzt. Der Ausschlag hierfür war zum einen, dass alle anderen Verfahren unnötige, zusätzliche Aktionen vom Kunden erfordern und zum anderen, dass die Anforderungstreue eine unverfälschte Momentaufnahme sein soll. Der Kompromiss hierbei ist, dass eine Anforderungsänderung direkt vor Projektende in der Regel nicht mehr erfüllt werden kann und die Anforderungstreue so leidet, ohne dass die Entwickler einen Einfluss darauf haben können. Dieser Kompromiss wird allerdings bewusst eingegangen. Sollte so etwas passieren, verlässt der Sachverhalt das Thema dieses Konzepts. Die Anforderungstreue soll ein möglichst klares und eindeutiges Maß sein, was zusätzlich für das ausgewählte Verfahren spricht.

3.1.4 Bewertung

Nachdem das Projekt beendet ist und dem Kunden das Produkt geliefert worden ist, wird er im Rahmen dieses Konzepts den Erfolg der fixierten Anforderungen bewerten. Wie dieses geschieht wird in diesem Subkonzept erkläutert.
Abbildung 3.7 gliedert diesen Schritt in das Gesamtkonzept ein.

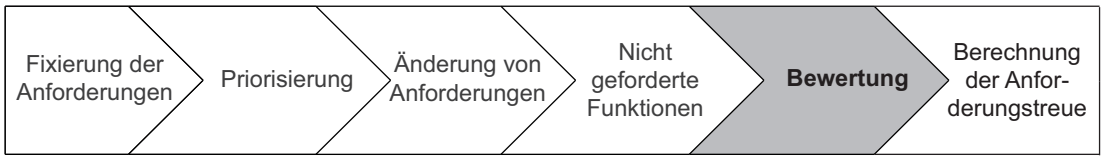

Abbildung 3.7: Eingliederung der Bewertung in den Gesamtprozess

Ziele

Dieses Subkonzept muss leisten, dass der Kunde seine subjektive Zufriedenheit mit einzelnen Anforderungen schnell, unkompliziert und möglichst genau erfassen und festhalten kann. Zusätzlich muss es möglich sein die Ergebnisse dieses Subkonzepts an eine Formel zur Berechnung der Anforderungstreue zu übergeben und zu berechnen.

5 Sterne Bewertung

Das Verfahren, welches für diese Arbeit ausgewählt worden ist, gibt dem Kunden die Möglichkeit jedes eingegebene Feature mit 1-5 Sternen zu bewerten. 1 Stern gilt dabei als minimale Zufriedenheit und 5 Sterne als maximale Zufriedenheit. Diese Bewertungsmethode ist im Internet auf diversen großen Seiten verbreitet, sodass nahezu jeder damit schon in Kontakt gekommen ist. Bekannte Seiten wie Amazon[2], Netflix[3] oder iTunes[4] benutzen dieses Verfahren zur Bewertung von Produkten durch Kunden. Abbildung 3.8 stellt dieses Verfahren graphisch dar.

Vorteile diese Methode sind:

- Leichtgewichtigkeit
  - Der Kunde kann durch einen einfachen Klick auf einen Stern (im implementieren Werkzeug bzw. Prototyp) seine Bewertung für ein Feature abgeben. Dieses ist ein anschaulicher und schneller Vorgang.
- Kein nötiges Spezialwissen
  - Diese Methode ist extrem verbreitet und bekannt. Es ist davon auszugehen, dass sehr viele potentielle Kunden mit diesem oder einem ähnlichen Verfahren bereits konfrontiert worden sind. Das Verfahren ist also nicht nur sehr einfach zu benutzen, sondern viele Kunden werden damit schon vertraut sein.
- In eine mathematische Formel übertragbar
  - Die Bewertung von 1 bis 5 kann man prinzipiell direkt als Faktoren für ei-

[2] http://www.amazon.com
[3] http://www.netflix.com
[4] http://www.apple.com/itunes

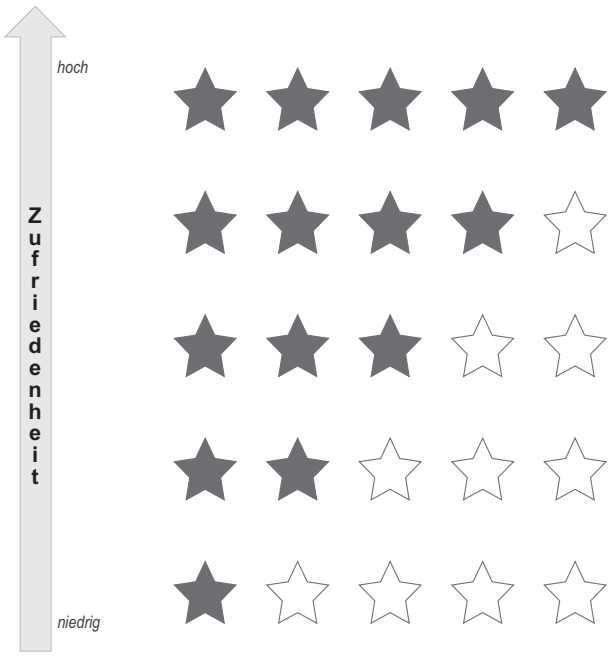

Abbildung 3.8: 5 Sterne Bewertungsmethode

ne spätere Verrechnung in eine mathematische Formel übertragen, um die Anforderungstreue zu berechnen.

- Es ist möglich teilweise erfüllte Features zu bewerten
  - Sollte ein Feature mehrere einzelne Funktionen beinhalten, dann kann man mit diesem Verfahren zum Beispiel 2-4 Sterne vergeben, wenn einige Funktionen des Features erfolgreich implementiert worden sind und andere nicht.

Nachteile:

- Das Verfahren neigt dazu, den Bewerter dazu zu verleiten extreme (1 oder 5 Sterne) Bewertungen zu vergeben. Zum Beispiel YouTube [5] hat früher ein solches Verfahren zum Bewerten benutzt und festgestellt, dass die Bewertungen nicht gleich verteilt waren, sondern 5 und 1 Sterne Bewertungen klar dominiert haben. Dieser Nachteil kommt allerdings zustande, da nicht jeder Nutzer eine Bewertung abgeben muss und viele Nutzer keine Motivation haben Bewertungen abzugeben [You09]. In Abbildung 3.9 sind die Anzahl der verschiedenen YouTube Bewertungen aus dem Jahr 2009 dargestellt.

Für diese Arbeit ist allerdings damit zu rechnen, dass dieser Nachteil nicht zum tragen kommt, da die Kunden von sich aus eine hohe Motivation mitbringen, die eingegebenen Features zu bewerten. Sie sind nicht einfach Konsumenten, sondern möchten vom Konzept eine möglichst passende Rückmeldung erhalten.

[5] http://www.youtube.com

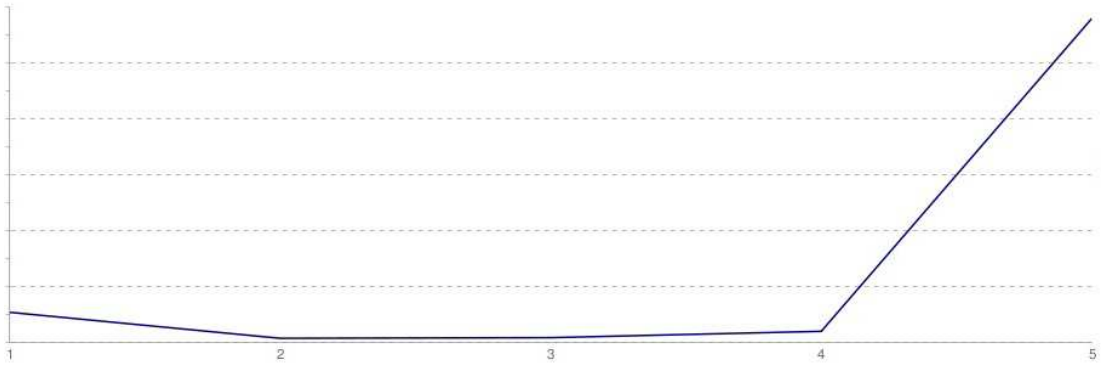

Abbildung 3.9: Verteilung von YouTube Bewertungen mit 5 Sterne Methode, Quelle: [You09]

Wechselwirkungen mit anderen Subkonzepten

Dieses Subkonzept hat essentiellen Einfluss auf das Gesamtkonzept. Aus den vom Kunden eingegebenen Werten berechnet sich letztendlich ein tatsächlicher Wert der Anforderungstreue. Die hier eingegebenen Werte werden an den nächsten Schritt des Subkonzepts weitergegeben und dort in irgendeiner Form in Elemente einer Formel umgewandelt.

Herausforderungen und wichtige Attribute

Dieses Subkonzept muss folgende Attribute aufweisen:

- Leichtgewichtigkeit
  - Der Kunde muss mit wenig Zeiteinsatz in der Lage sein Erfolgsbewertungen einzugeben.
- Berechenbarkeit in 'Berechnung der Anforderungstreue'
  - Das Subkonzept muss die eingegebenen Bewertungen in einer Form weitergeben können, die in einer mathematischen Formel verarbeitet werden kann.
- Kein nötiges Spezialwissen
  - Die Bewertung wird vom Kunden durchgeführt, also muss sie möglichst einfach und intuitiv durchzuführen sein.

Vergleich verschiedener Verfahren / Verwandte Arbeiten

Bewertung per Faktor
Dieses Verfahren geht analog zum gleichnamigen Verfahren zur Priorisierung, wie

oben vorgestellt, vor. Der Kunde kann die Erfüllung jedes Features von 0% bis 100% bewerten.
Vorteile dieser Methode:

- Eine sehr feine Abgrenzung der Bewertung ist möglich.
- Der Faktor für die spätere Berechnung der Bewertung in der Formel für die Anforderungstreue ergibt sich aus der relativen Gewichtung, die sich aus den vergebenen Faktoren errechnet.
- Es ist möglich, teilweise erfüllte Features zu bewerten.

Dagegen stehen folgende Nachteile:

- Eine Abgrenzung zwischen verschiedenen Bewertungen ist sehr schwierig - die Entscheidung ob eine Priorität zum Beispiel 70% oder 71% ist, ist nahezu unmöglich.
- Die sehr feinen Auswahlmöglichkeiten bedeuten eine höhere Zeitinvestition für den Kunden.
- Diese Methode ist sehr abstrakt und wenig anschaulich, daher steigt das Risiko für falsche Eingaben.

Binäre Bewertung
Nachdem YouTube die Probleme des 5 Sterne Verfahrens erkannt hat [You09], wurde dort ein einfaches binäres System eingeführt. Entweder man markiert ein Video als positiv oder eben nicht. Ähnlich binär gehen auch zum Beispiel soziale Seiten wie Facebook [6] vor.
Bei diesem Verfahren hat der Kunde die Möglichkeit ein Feature entweder positiv mit 'erfüllt' oder negativ mit 'nicht erfüllt' zu bewerten. Zwischenstufen gibt es nicht.
Dieses sehr vereinfachte Verfahren bietet folgende Vorteile:

- Leichtgewichtigkeit
  - Diese sehr einfache Methode bedeutet minimalen Zeitaufwand für den Kunden.
- Kein nötiges Spezialwissen
  - Diese Methode ist sehr einfach und verbreitet. Ein Kunde kann sie ohne Einarbeitung verwenden.

Nachteile:

- Es ist nicht möglich teilweise erfüllte Features zu bewerten
  - Sollte ein Feature mehrere einzelne Funktionen beinhalten, von denen nur einige erfolgreich implementiert sind, dann kann der Kunde trotzdem nur volle oder keine Zufriedenheit bewerten.

[6] http://www.facebook.com

Begründung der Auswahl

In Tabelle 3.4 ist eine Übersicht der Verfahren in Bezug auf wichtige Attribute dieses Subkonzepts dargestellt.

	5 Sterne	Faktor	Binär
Leichtgewichtigkeit	✓	O	✓
Berechenbarkeit	✓	✓	O
Kein nötiges Spezialwissen	✓	O	✓

Tabelle 3.4: Vergleich der Methoden zur Bewertung der Zufriedenheit

Leichtgewichtigkeit: Prinzipiell sind alle drei Verfahren leichtgewichtig. Insbesondere die binäre und die 5 Sterne Bewertung. Verlangt man vom Kunden eine genauer prozentuale Bewertung, ist er in jedem Falle gezwungen sich mehr Zeit zu investieren, da die Abgrenzung zwischen den Werten feiner und somit schwieriger ist.
Berechenbarkeit: Das 5 Sterne Verfahren und die Vergabe eines prozentualen Faktors eignen sich beide sehr gut dafür, diese Bewertungen in eine mathematische Formel weiterzugeben. Somit ist es möglich die vom Kunden eingegebenen Rohdaten direkt für die Berechnung zu benutzen, ohne dass das Konzept selbst Faktoren bestimmen muss. Das binäre Verfahren ist dazu zwar ebenfalls in der Lage, hat aber das große Problem, dass teilweise erfüllte Features entweder ganz oder gar nicht zufrieden bewertet werden müssen. Somit gäbe es in diesem Fall eine Verfälschung der Anforderungstreue.
Kein nötiges Spezialwissen: Alle drei Verfahren benötigen keine intensive Einarbeitung, allerdings ist es insbesondere beim binären und 5 Sterne Verfahren für den Kunden leichter Bewertungen abzugeben, da die Abgrenzungen grob sind.

Das ausgewählte Konzept in dieser Arbeit ist das 5 Sterne System. Dieses Verfahren hat im Bezug auf das Ziel dieser Arbeit nahezu keine Schwachstelle: es ist flexibel genug um allen möglichen Situationen (nicht implementierte Features, teilweise implementierte Features, gut implementierte Features) verarbeiten zu können und gleichzeitig einfach genug, um den Kunden mit möglichst wenig Zeitaufwand anschauliche Bewertungen vergeben zu lassen.

3.1.5 Nicht geforderte Funktionen

Wie geht man mit gelieferten Funktionen der Software um, die sich in keiner Anforderung des Kunden wiederfinden? Mit dieser Frage beschäftigt sich dieses Subkonzept. Abbildung 3.10 gliedert diesen Schritt in das Gesamtkonzept ein.

Ziele

Die Ziele dieses Subkonzepts sind zum einen Leichtgewichtigkeit: der Kunde soll möglichst wenig Eingaben machen müssen und die Durchführung muss möglichst intuitiv

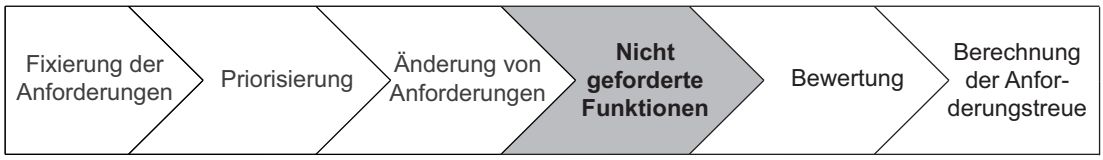

Abbildung 3.10: Eingliederung der nicht geforderten Funktionen in den Gesamtprozess

sein. Dabei soll der Kunde trotzdem die Möglichkeit haben einen Einfluss auf den Wert der Anforderungstreue auszuüben, wenn die gelieferte Software Funktionen enthält, die sich in keiner fixierten Anforderung wiederfinden.

Negative Features

Die ausgewählte Methode, um mit diesem Problem umzugehen, sind negative Features.

Definition 2 (Negativ Feature) *Eine Eigenschaft oder Funktionalität eines Softwareprogramms oder dessen Umfelds (zum Beispiel Dokumentation), welche sich nicht in den fixierten Kundenanforderungen wiederfindet.*

Der Kunde ist in der Lage nach Projektende Features einzugeben, welche sich nicht in seinen Anforderungen wiederfinden. Zu diesen negativen Features kann er keine Priorität und keine Bewertung abgeben – der Einfluss auf die Formel der Anforderungstreue wird vom Konzept vorgegeben. Das verringert zum einen den Zeitaufwand, den der Kunde investieren muss, und zum anderen garantiert es einen konsistenten Umgang mit Situationen.
Vorteile dieses Verfahrens sind:

- Leichtgewichtigkeit
  - Der Kunde muss nichts weiter tun, als eine Funktion beschreiben.
- Kein nötiges Spezialwissen
  - Der Kunde muss sich nach einer einfachen Eingabe keine Gedanken um die Berechnung oder Bewertung machen, diese Aufgabe wird komplett vom Subkonzept übernommen.

Nachteile:

- Der Kunde muss in der Lage sein Funktionen zu erkennen, die sich nicht in seinen Anforderungen wiederfinden.

Wechselwirkungen mit anderen Subkonzepten

Dieses Subkonzept hat Auswirkungen auf den letzten Schritt: die explizite Berechnung eines Wert für die Anforderungstreue. Je nachdem, wie sich dieses Subkonzept verhält gibt es verschiedene Auswirkungen auf den Wert der Anforderungstreue.

Herausforderungen und wichtige Attribute

Folgende Attribute muss dieses Subkonzept besitzen:

- Leichtgewichtigkeit
  - Der Kunde muss ohne viel Zeitaufwand in der Lage sein, Einfluss auf den Wert der Anforderungstreue zu nehmen, wenn Funktionen geliefert worden sind, die sich in keiner fixierten Anforderung wiederfinden.
- Kein nötiges Spezialwissen
  - Da wir potentielle Kunden auf kein bestimmtes Profil reduzieren können, muss dieses Subkonzept möglichst intuitiv ohne viel Einarbeitung benutzbar sein.

Vergleich verschiedener Verfahren

Alternativ wurde im Rahmen dieser Arbeit folgende Methode untersucht.

Gesamtwert der Unzufriedenheit

Bei diesem Verfahren gibt der Kunde nach Projektablauf einen bestimmten Wert (zum Beispiel 10%) auf einer bestimmten Skala (zum Beispiel 0%-100%) an. Dieser Wert ist seine Einschätzung davon, wie viel von der gelieferten Software sich nicht in seinen fixierten Anforderungen wiederfindet.

- Leichtgewichtigkeit
  - Der Kunde muss lediglich einen Wert angeben, was isoliert betrachtet kaum Zeitaufwand bedeutet.

Nachteile:

- Spezialwissen zum Teil erforderlich
  - Einen solchen Wert fundiert zu beurteilen benötigt sehr viel Wissen und Überblick.
- Diese Methode führt zu einer relativ instabilen Formel
  - Dieser Faktor hätte einen extremen Einfluss auf den Wert der Anforderungstreue. Nicht fundierte Eingaben können dazu führen, dass das Projekt vom Gesamtkonzept massiv fehl eingeschätzt wird.

Begründung der Auswahl

In Tabelle 3.5 ist eine Übersicht beider Verfahren in Bezug auf die geforderten Attribute dieses Subkonzpts dargestellt.

	Negativ Features	Gesamtwert der Unzufriedenheit
Leichtgewichtigkeit	✓	✓
Kein nötiges Spezialwissen	✓	O

Tabelle 3.5: Vergleich der Methoden zum Umgang mit nicht geforderten Funktionen

Leichtgewichtigkeit: das Verfahren der Negativ Features ist für den Kunden unkompliziert und zeitsparend. Er muss lediglich Funktionen beschreiben die nicht seinen Anforderungen entsprechen. Allerdings kann es, je nach gelieferter Software, sein dass er mehrere Eingaben machen möchte, wenn verschiedene nicht geforderte Funktionen existieren. Beim zweiten Verfahren, der Eingabe eines Gesamtwertes, ist der Ablauf sogar noch leichtgewichtiger, da lediglich eine Aktion vom Kunden ausgeführt werden muss, unabhängig davon, wie viele nicht geforderte Funktionen existieren.
Kein nötiges Spezialwissen: Negativ Features benötigen keinerlei Spezialwissen und keine fundierte Einschätzung seitens des Kunden. Die gesamte weitere Berechnung wird allein vom Konzept übernommen, für den Kunden unsichtbar. Die Eingabe eines fundierten Gesamtwertes allerdings benötigt ein hohes Maß an Genauigkeit, um keinen falschen Einfluss auf den Wert der Anforderungstreue zu nehmen. Er kann hier quasi beliebigen negativen Einfluss auf die Anforderungstreue nehmen.

Aufgrund oben genannter Gründe wird für dieses Konzept das Verfahren der Negativ Features ausgewählt. Diese Methode ist zum einen leichtgewichtig und einfach zu benutzen, zum anderen hat sie einen robusteren und konsistenteren Einfluss auf den Wert der Anforderungstreue, als das vorgestellte, alternative Verfahren.

3.1.6 Berechnung der Anforderungstreue

In diesem letzten Subkonzept werden schließlich die Ergebnisse alle vorherigen Schritte gesammelt und dazu benutzt, um einen Wert für die Anforderungstreue des gegebenen Projekts zu berechnen. Abbildung 3.11 gliedert dieses in den Gesamtprozess ein.

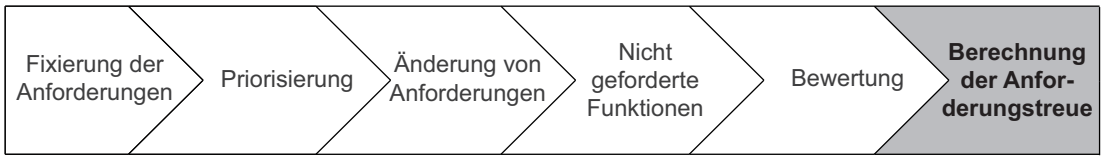

Abbildung 3.11: Eingliederung der Berechnung der Anforderungstreue in den Gesamtprozess

Ziele

Das Ziel dieses Subkonzepts ist es, einen expliziten Wert auszugeben, den der Kunde nachvollziehen kann und der sich natürlich auf die vorherigen Eingaben stützt.

Entwicklung der Formel

Wie schon in der Einleitung dieser Arbeit erwähnt, wurde in 'Requirements Compliance as a Measure of Project Success' [SLPK13] bereits eine Formel für die Anforderungstreue aufgestellt, siehe Formel 2.1. Alle erfüllten Anforderungen gehen mit dem Faktor 1 positiv in die Formel ein, während nicht geforderte Anforderungen mit dem Faktor 0,5 davon subtrahiert werden. Diese Subtraktion schauen wir uns genauer an: sollte der Fall eintreten, dass $1 \cdot S$ kleiner ist als $0,5 \cdot I$, dann wird der Wert der Anforderungstreue negativ. Denkt man nun an die Anschauung, dass die Anforderungstreue die Größe der Schnittmenge der Kundenanforderungen C und implementierten Anforderungen I ist (siehe Abbildung 3.12), dann wird unklar, was ein negativer Wert im Vergleich zum Wert 0 zu bedeuten hat. Bleibt man bei der Mengenanschauung, dann bedeutet eine Substraktion, dass die Menge der erfolgreich implementierten Anforderungen $S = C \cap I$ verkleinert wird. Richtig wäre allerdings, wie in der Grafik anschaulich zu sehen ist, dass die implementierten Anforderungen einfach im Bereich außerhalb von C liegen.

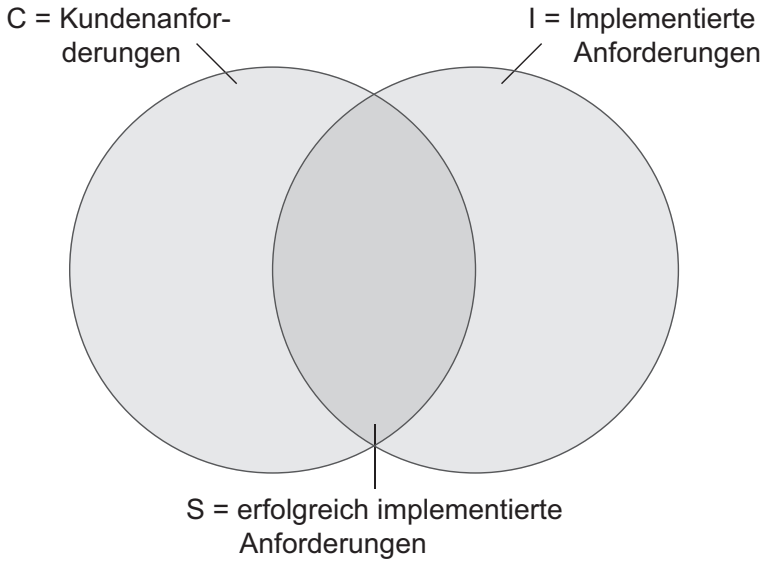

Abbildung 3.12: Anforderungstreue als Schnittmenge

Die in dieser Arbeit entwickelte Formel für die Anforderungstreue wird daher auf eine Subtraktion verzichten. Dieses Problem wird in der Formel so dargestellt werden, dass die maximale Anzahl an erreichbaren Punkten steigt und es einen Teil unerfüllbare Punkte geben wird, wenn nicht geforderte Anforderungen implementiert werden.

Betrachten wir, welche Informationen uns die vorherigen Schritte bis hier liefern:

- Eine Menge von Features mit folgenden zusätzlichen Informationen
  - Priorität
    - MUST, SHOULD oder COULD
  - Bewertung
    - 1-5 Sterne
- Eine Menge von negativen Features ohne zusätzliche Informationen wie Priorität oder Bewertung

Es müssen nun die Prioritäten und Bewertungen in mathematische Faktoren umgewandelt werden. Die Prioritäten werden zunächst wie folgt mit abstrakten Faktoren verbunden:

- MUST = 10
- SHOULD = 5
- COULD = 1

Dieses wird letztendlich zur Folge haben, dass ein ein MUST genau den Wert von zwei SHOULDs hat und ein SHOULD wiederum den Wert von fünf COULDs. Diese Werte haben sich im Laufe einer Studie (siehe unten) als Werte herausgestellt, die die subjektive Kundenempfindung realistisch widerspiegeln. Verändert man diese Zuordnung der Werte so hat dieses natürlich Auswirkungen auf den letztendlich errechneten Wert, allerdings geht es in diesem Konzept im Wesentlichen um eine schnelle und relativ gute Einschätzung: Ob ein Wert von 75% oder 72% errechnet wird hat keinen großen Unterschied, aber es ist bei beiden Werten klar zu erkennen, dass sie geringer als zum Beispiel 95% sind. Die perfekte Genauigkeit gibt es nicht, da im Umgang mit Anforderungen eben immer Subjektivität im Spiel ist.
Die Sternebewertung in eine Formel zu übernehmen geschieht sehr direkt: es existiert eine maximale Anzahl an zu erreichenden Sternen und eine tatsächlich bewertete Anzahl von erfolgreichen Sternen. Teilt man die erreichten Sterne durch die maximale Anzahl an möglichen Sternen ergibt dies einen Wert von 0 bis 1. Genauso wird die Formel der Anforderungstreue vorgehen, mit zusätzlicher Beachtung aller anderen Subkonzepte.
Das letzte fehlende Element für die Formel sind die negativen Features. Diese werden wie oben erwähnt nichts vom bisher positiv erreichten Wert subtrahieren, sondern sie erhöhen die maximale Anzahl an erreichbaren Sternen um 5. Diese 5 Sterne können nicht positiv bewertet werden (sie waren nicht gefordert) und senken so den Wert der Anforderungstreue.

Letztendlich ergibt sich nach diesem Konzept also folgende Formel:

$$RC = \frac{\sum_{Features}(prio(Feature) \cdot sterne(Feature))}{\sum_{Features}(prio(Feature) \cdot 5) + 5 \cdot \#NegativFeatures}$$

prio(Feature) = Prioritätsfaktor des aktuellen Features, sterne(Feature) = Sternebewertung des aktuellen Features

(3.1)

Die Anforderungstreue ist nach dieser Formel ein wert von 0 bis 1. 0 bedeutet dabei, dass die Anforderungen des Kunden gar nicht erfüllt worden sind und 1 bedeutet sie sind maximal erfüllt worden, ohne dass nicht geforderte Funktionen implementiert worden sind.
Die Interpretation dieses Wertes, ab welchem Wert ein Kunde zum Beispiel zufrieden sein wird oder nicht, kann nicht fest vorgegeben werden. Vielmehr ist eine Fallunterscheidung vom jeweiligen Projekt und Kunden nötig. Zwei Aussagen, die allerdings für jedes Projekt gelten sind: ist die Anforderungstreue 0, wurden keine Kundenanforderungen erfüllt und das Projekt ist mit Sicherheit ein massiver Misserfolg. Ist die Anforderungstreue 1 (bzw. 100%), wurden genau alle Kundenanforderungen erfüllt und das Projekt ist ein beispielhafter Erfolg.

3.2 Übersicht

Um das Gesamtkonzept nochmal in seiner Gänze zusammenzufassen, ist hier ein Überblick gegeben.

1. Der Kunde gibt beliebig viele Features ein.
   - Im Wesentlichen ist dieses ein Fließtext, in den der Kunde nahezu schreiben kann, was er möchte.
2. Alle Features werden vom Kunden nach MoSCoW Methode priorisiert.
   - Mögliche Stufen sind MUST (maximal wichtig), SHOULD und COULD (minimal wichtig).
3. Sollte es Änderungen an den Anforderungen geben kann er alle Features zu jedem Zeitpunkt beliebig verändern.
4. Nach Lieferung der Software bewertet der Kunde alle eingegebenen Features mittels der 5 Stern Methode.
   - Für jedes Feature hat er die Möglichkeit seine Zufriedenheit mit der Implementierung 1-5 Sterne zu vergeben.
5. Der Kunde pflegt nicht geforderte Funktionen in Form von negativen Features ein.
   - Im Wesentlichen ist dieses ein Fließtext, in den der Kunde nahezu schreiben kann, was er möchte.
6. Die oben entwickelte Formel liefert als diesen Informationen einen Wert für die Anforderungstreue.

- Mögliche Werte sind 0 bis 1 (bzw 0% bis 100%).

3.3 Beispiel

In diesem Abschnitt wird das Konzept anhand eines Beispiels verdeutlicht. Das folgend aufgebaute Beispiel erhebt keinen Anspruch auf Vollständigkeit, es ist lediglich als anschauliches Beispiel zum Verständnis des Konzepts zu verstehen.

3.3.1 Projekt: Requirement Compliance Assessor

Der Kunde möchte den Requirement Compliance Assessor: ein Webservice zur Durchführung eines Konzepts zur Erfassung der Anforderungstreue.

Features eingeben
Folgende Features werden vom Kunden eingegeben:

#1: Intuitive Bedienung	#2: Features eingeben
Für den Kunden ist die Bedienung intuitiv und schnell	Zu jedem Projekt kann der Kunde Features eingeben, die Anforderungen widerspiegeln

#3: Mehrere Projekte	#4: Bewertung
Der Webservice muss in der Lage sehr mehrere Projekte zu verwalten	Der Kunde muss in der Lage sein alle Features mittels 5 Sterne Bewertungsmethode zu bewerten

#5: Editierbarkeit	#6: Ergebnisberechnung
Projekte und Features sind auch im Nachhinein noch beliebig veränderbar	Für jedes Projekt existiert eine Auswertungsseite auf der aus den eingegebenen Informationen ein Wert für die Anforderungstreue präsentiert wird

Priorisierung
Der Kunde priorisiert die Features wie folgt.

Nicht geforderte Funktionen
Der Kunde findet eine Funktion, die er nicht gefordert hat.

#1: Intuitive Bedienung	#2: Features eingeben
MUST	MUST

#3: Mehrere Projekte	#4: Bewertung
SHOULD	MUST

#5: Editierbarkeit	#6: Ergebnisberechnung
COULD	MUST

#N1: E-Mail Funktion
Es existiert eine nicht geforderte E-Mailfunktion, die automatisch bei Änderungen an einem Projekt den Projektbesitzer benachrichtigt.

Bewertung
Der Kunde vergibt für die fixierten Features folgende Bewertungen:

#1: Intuitive Bedienung	#2: Features eingeben
4/5	5/5

#3: Mehrere Projekte	#4: Bewertung
5/5	3/5

#5: Editierbarkeit	#6: Ergebnisberechnung
2/5	5/5

Berechnung der Anforderungstreue
Die eingegebenen Funktionen werden nun der Formel 3.1 übergeben, daraus ergibt sich folgende Berechnung:

$$RC = \frac{10 \cdot 4 + 10 \cdot 5 + 5 \cdot 5 + 10 \cdot 3 + 1 \cdot 2 + 10 \cdot 5}{(10 \cdot 5 + 10 \cdot 5 + 5 \cdot 5 + 10 \cdot 5 + 1 \cdot 5 + 10 \cdot 5) + 5 \cdot 1} = 0,84$$

Dieses Beispielprojekt erreicht einen Wert von 0,84 beziehungsweise 84% Anforderungstreue.

3.4 Prototyp Tests und Benutzerstudie

Für diese Arbeit wurde ein Prototyp entwickelt, welcher das entwickelte Konzept innerhalb eines Webservices abbildet. Mit diesem Prototyp wurden Tests an potentiellen Kunden durchgeführt.

3.4.1 Interviewkonzept

Das den Tests zugrundeliegende Interviewkonzept orientiert sich an der Goal-Question-Metrik [Bal08]. Zunächst wurden die zu untersuchenden Aspekte herausgearbeitet. Zu diesen anschließend Qualitäts- und Einflussfaktoren erarbeitet und Ausgangshypothesen aufgestellt. Aus diesen Informationen wurden die Interviewfragen entwickelt und Ableitungen aus möglichen Antworten erarbeitet.

Aspekt	Leichtgewichtigkeit
Qualitätsfaktoren	Nötige Zeit zur Eingabe
Einflussfaktoren	Nötige Menge der Texteingabe Nötiges Spezialwissen Verständlichkeit der Aufgabe
Ausgangshypothesen	Die gesamte Eingabe aller Features sollte nicht länger als 30 Minuten dauern
Interviewfragen	Wie viel Zeit haben sie haben Sie benötigt? Ist dieser Zeitraum für sie subjektiv leichtgewichtig? Wenn nein, wie hätte die benötigte Zeit minimiert werden können?
Ableitungen aus den Antworten	Sollte die benötigte Zeit mehr als 30 Minuten betragen, dann muss überprüft werden, ob das Konzept nicht verständlich genug ist. Sollte es Probleme bei formalen Abläufen geben, muss eine genauere Anleitung erstellt werden, wie der Kunde die Eingaben tätigen soll.

Tabelle 3.6: Interviewkonzept zur Leichtgewichtigkeit

Aspekt	Qualität der Anleitung
Qualitätsfaktoren	Formulierung und Anzahl der erfassten Features Benötigte Zeit
Einflussfaktoren	Benötigtes Spezialwissen vom Kunden Granularität und Länge der Anleitung
Ausgangshypothesen	Je besser die Anleitung, desto reibungsloser wird der gesamte Prozess laufen. Sowohl die benötigte Zeit als auch die Qualität der eingegebenen Daten werden maßgeblich von der Anleitung beeinflusst.
Interviewfragen	Hatten sie beim Durchführen der Tests Probleme mit dem Prozessablauf? Hatten sie offene Fragen, die sie zum Fortfahren der Tests unbeantwortet lassen mussten?
Ableitungen aus den Antworten	Wenn es bei der Bewertung Probleme gibt, dann sollte ein Feature noch unterteilt in zum Beispiel Unterfunktionen damit besser zu entscheiden ist, zu welchem Grad ein Feature erfüllt ist. Die Definition der Features würde in diesem Fall angepasst werden.

Tabelle 3.7: Interviewkonzept zur Qualität der Anleitung

Aspekt	Eignung der Featuredefinition
Qualitätsfaktoren	Vollständigkeit der für den Kunden relevanten erfassten Anforderungen Genauigkeit in der Bewertung
Einflussfaktoren	Definition und Verständnis des Featurebegriffs Möglichkeit zu entscheiden, zu welchem Grad ein Feature funktioniert Anzahl der verschiedenen Funktionen in einem Feature Anzahl und Granularität der erfassten Features
Ausgangshypothesen	Es sollten möglichst vom Umfang gleichgroße Features erfasst werden. Es sollten nicht zu viele (<20) Features erfasst werden. Ein Feature wird mehrere Einzelfunktionen besitzen, dies könnte die Gesamtbewertung des Features erschweren.
Interviewfragen	Hatten Sie bei der Darstellung ihrer Anforderungen in Form von Features Probleme? Fiel es ihnen leicht Features voneinander abzugrenzen? Hat die erzwungene Benutzung der benutzen Featuredefinition sie eingeschränkt? Wenn ja, was konnten Sie nicht ausdrücken? Könnten Sie die Features leicht auf auf Erfüllung bewerten? Wäre es schwierig bei der Bewertung, wenn nur ein Teil des Features implementiert ist, ein anderer Teil aber nicht?
Ableitungen aus den Antworten	Sollte es Probleme geben Anforderungen als Features darzustellen, muss die Definition angepasst werden. Insbesondere die offene Definition bei der Formulierung sollte überprüft werden. Wenn die Anzahl oder Granularität unpassend ist, dann muss die Anleitung zur Erfassung der Features verfeinert werden.

Tabelle 3.8: Interviewkonzept zur Featuredefinition

Aspekt	Eignung der Priorisierungsmethode
Qualitätsfaktoren	Genauigkeit in der Priorisierung
Einflussfaktoren	Definition und Verständnis des Verfahrens Anzahl der verschiedenen Prioritätsstufen
Ausgangshypothesen	Es sollte ohne Probleme möglich sein sofort zu entscheiden, welche Prioritätsstufe einem Feature zugeordnet wird. Der Zeitaufwand wird voraussichtlich relativ gering sein.
Interviewfragen	Hatten Sie bei der Priorisierung von Features Probleme? Fiel es ihnen leicht die Prioritätsstufen voneinander abzugrenzen? Hätten Sie sich mehr/weniger? mögliche Prioritätsstufen gewünscht?
Ableitungen aus den Antworten	Sollte es Probleme bei der Priorisierung geben, muss das Verfahren angepasst werden. Sollte es Probleme dabei gegeben haben die Prioritätsstufen voneinander abzugrenzen oder sollten sich Kunden mehr/weniger Stufen gewünscht haben, dann sollte ein Verfahren getestet werden, welches diese Funktionalitäten bietet.

Tabelle 3.9: Interviewkonzept zur Priorisierungsmethode

Aspekt	Eignung der Bewertungsmethode
Qualitätsfaktoren	Genauigkeit in der Bewertung der Features Nötiger Zeitaufwand
Einflussfaktoren	Definition und Verständnis des Verfahrens Anzahl der verschiedenen Bewertungsstufen
Ausgangshypothesen	Es sollte ohne Probleme möglich sein sofort zu entscheiden, welchen Erfolgswert einem Feature zugeordnet wird. Der Zeitaufwand wird voraussichtlich relativ gering sein, da das Verfahren intuitiv benutzbar ist.
Interviewfragen	Hatten Sie bei der Bewertung von Features Probleme? Fiel es ihnen leicht die Bewertungsstufen (5 Sterne) voneinander abzugrenzen? Hätten Sie sich mehr/weniger? mögliche Bewertungsmöglichkeiten gewünscht?
Ableitungen aus den Antworten	Sollte es Probleme bei der Bewertung geben, muss das Verfahren angepasst werden. Sollte es Probleme dabei gegeben haben die Bewertungsstufen voneinander abzugrenzen oder sollten sich Kunden mehr/weniger Stufen gewünscht haben, dann sollte ein Verfahren getestet werden, welches diese Funktionalitäten bietet.

Tabelle 3.10: Interviewkonzept zu Bewertungsmethode

3.4.2 Ergebnis der Interviews und Tests am Prototypen

Insgesamt wurden sechs Projekte mit dem entwickelten Prototypen getestet. Zunächst seien einige Rohdaten der Ergebnisse präsentiert:

Durchschnittl. Anzahl Features	9,33
Durchschnittl. Anzahl MUST	4,66
Durchschnittl. Anzahl COULD	3,66
Durchschnittl. Anzahl SHOULD	1
Durchschnittl. Anzahl neg. Features	0,17
Durchschnittl. Bewertung	ca. 4,5
Durchschnittl. Anforderungstreue	89%

Tabelle 3.11: Rohdaten zu den Tests am Prototypen

Bevor die Tests zu den einzelnen Subkonzepten erläutert werden, ist zu sagen, dass alle Tests sehr erfolgreich abliefen und die Kunden den Prozessen gut folgen konnten und sich mit dem Ausgegebenen wert identifizieren konnten.

Anforderungen als Features
Bei einem Projekt gab es zunächst Unklarheiten, was die Definition genau bedeutet. Diese Unklarheiten haben sich jedoch schnell geklärt. In allen anderen fünf Fällen lief die Eingabe der Anforderungen reibungslos und intuitiv. Die durchschnittliche Zeit für die Eingabe eines Features belief sich auf etwa eine Minute. Bei durchschnittlich 9,33 eingegebenen Features war die reine Zeit der Anforderungsfixierung dementsprechend unter 10 Minuten, was als großer Erfolg für das Subkonzept und den Prototyp zu bewerten ist.

Priorisierung
Die Priorisierung lief bei allen Tests intuitiv ohne Nachfragen. Alle Kunden haben die Stufen gut voneinander Abgrenzen können und Features mit allen 3 Prioritätsstufen angelegt. Besonders zu erwähnen ist hierbei, dass jedes Projekt genau ein COULD Feature aufzuweisen hatte und jedes dieser Features Bezug zur Benutzeroberfläche hatte.
Generell war dieses Subkonzept ebenfalls sehr erfolgreich, da keine wesentliche Zeitinvestition nötig war und alle Features problemlos priorisiert worden sind.

Nicht geforderte Funktionen
Dieses Subkonzept ist den Kunden nicht so intuitiv von der Hand gegangen wie die vorherigen. Es wurde nur in einem Projekt ein einziges negatives Feature angelegt. Ein Kunde hatte auch schlichtweg keine Lust das Programm auf nicht geforderte Funktionen zu durchsuchen, da diese in seinen Augen unnötigen Zeitaufwand bedeutet hätte. In allen anderen Fällen gab es keine nicht geforderten Funktionen im Programm.

Prinzipiell ist dieses Konzept dennoch als erfolgreich zu betrachten, denn nach Rücksprache wäre jeder Kunde in der Lage gewesen, potentiell nicht geforderte Funktionen in den Prototyp einzugeben. Im Projekt mit einem negativen Feature wurde der Kunde mit der alternativen Methode (siehe oben) befragt, ob er einschätzen kann, wie viel Prozent vom Programm ca. nicht gefordert war. Diese Frage konnte er allerdings 'nicht ohne zu raten' beantworten.

Bewertung
Die Bewertungsmethode hat durchweg Begeisterung hervorgerufen, da sie von allen Testteilnehmern als sehr einfach wahrgenommen worden ist. Alle Kunden waren schon vorher mit den 5 Stern Methode vertraut und konnten diese intuitiv richtig benutzen.
Dieses Subkonzept wurde am positivsten wahrgenommen und ist als sehr erfolgreich zu bewerten.

Berechnung der Anforderungstreue
Die ursprüngliche Version der Berechnung hat zu Ergebnissen geführt, die nicht die subjektive Empfindung der Kunden entsprochen hat. Zunächst waren die Prioritätsstufen MUST, COULD und SHOULD auf den Faktoren 1, 3 und 5 zugeordnet. Dies hat dazu geführt, dass MUST Features in Relation zu den anderen Stufen nicht so viel Ausschlag hatte, wie die Kunden gewünscht haben. Nach einigen Iterationen wurden die Faktoren 1, 5 und 10 gefunden – damit waren die Kunden subjektiv zufrieden mit den Werten für die Anforderungstreue.
Nach Anpassung der Faktoren ist dieses Konzept ebenfalls positiv aufgenommen worden und somit erfolgreich.

4 Diskussion

In diesem Kapitel werden die zuvor entwickelten Konzepte kritisch betrachtet und eventuelle Schwächen diskutiert. Die Grenzen des Konzepts werden beleuchtet und insbesondere die extrem zentrale Rolle des Kunden wird kritisch hinterfragt.

4.1 Subkonzepte

4.1.1 Fixierung der Anforderungen

Obwohl dem Begriff des Features eine explizite Definition zugrunde liegt, bleibt der Umgang damit doch bis zu einem gewissen Grad schwammig. Dem Kunden sind bei der Eingabe prinzipiell kaum Grenzen gesetzt, sodass er zwar auf der einen Seite genug Freiheiten hat, um jegliche Anforderungen auszudrücken, zum anderen aber birgt dieses einige Risiken. Der Kunde könnte Eingaben auf eine Art machen, die ihm selbst eine Bewertung in späteren Subkonzepten erschweren. Gibt er zum Beispiel alle seine Anforderungen in ein einziges (langes) Feature ein, so wird er für sein Projekt auch nur ein einziges Feature bewerten können. Gibt er sehr viele Features ein, dann bedeutet das für ihn einen unnötig hohen Zeitaufwand, da er alle diese Features priorisieren und bewerten muss.
Dadurch, dass keine explizite Erfolgsbedingung festgehalten wird, sondern die gelieferte Funktion letztendlich mit Fließtext verglichen wird, bleibt die Erfolgsbewertung ebenfalls schwammig. Bei bestimmten Fällen könnte es Interpretation vom fixierten Text benötigen, um ein Feature als erfolgreich oder nicht erfolgreich zu bewerten.
Letztendlich existieren diese Probleme zwar, aber sie haben voraussichtlich wenig Auswirkungen. Ein Kunde gibt seine eigenen Informationen ein und bewertet diese – es ist also zumindest jedes Projekt in sich konsistent bewertet. Eine gewisses Fingerspitzengefühl wird im Umgang mit Anforderungen immer nötig sein, auch mit diesem Konzept.
Ein weiterer wichtiger Vorteil dieses Subkonzepts ist die schiere Flexibilität des Featurebegriffs. Sollte das Konzept beispielsweise auf ein Projekt angewendet werden, für das bereits Anforderungen niedergeschrieben sind, zum Beispiel in einer Excel Datei oder als Story Cards während eines agilen Projekts, dann ist es möglich pragmatisch die bereits geschriebenen Anforderungen direkt als Features zu übernehmen, ohne weitere Anpassungen zu machen. Im agilen Fall hätte man dann ein Feature pro Story Card, was vielleicht nicht dem ursprünglichen Sinne des Konzepts entspricht, aber praktisch – wo oft nicht alles nach Plan und Lehrbuch geschehen kann – Vorteile bieten könnte.

4.1.2 Priorisierung

Die Priorisierung findet mittels MoSCoW Methode statt. Die drei Prioritätsstufen MUST, COULD, SHOULD müssen vom Konzept Faktoren zugeordnet werden, um sie später in einer mathematischen Formel zu verrechnen. Dieses ist für den Kunden allerdings unsichtbar und er hat keinerlei Einfluss darauf. Verschiedene Kunden könnten jedoch verschiedene Auffassungen haben, wie diese drei Stufen in Relation zueinander stehen.
Zusätzlich haben die zugeordneten Faktoren extremen Einfluss auf die Gesamtberechnung der Anforderungstreue. Nicht fundiert gewählte Faktoren können zu massiv verfälschten Ergebnissen führen. Dieses Problem ist in den Prototyptests im vorherigen Kapiteln deutlich geworden, so dass die benutzten Faktoren angepasst werden mussten. Je nach Kunde kann es also sein, dass der Wert der Anforderungstreue von seinem subjektiven Empfinden abweicht, weil diese Faktoren vom Konzept vorgegeben sind.

Randfallbetrachtung

Betrachten wir insbesondere das Verhalten bei Randfällen. Zur Erinnerung: die zugeordneten Faktoren für die Stufen sind MUST = 10, SHOULD = 5, COULD = 1.
Sollte ein Kunde zum Beispiel 20 COULD Features anlegen, kein SHOULD und nur ein MUST Feature, dann gehen die wichtigsten Anforderungen des Kunden nur zu einem drittel in die Berechnung ein. Sollte dieses MUST Feature nun nicht erfolgreich implementiert werden, aber alle unwichtigeren COULD Features, so wäre trotzdem noch ein relativ hoher Wert für die Anforderungstreue zu erreichen.
Der Gegenteilige Randfall (sehr viele MUST Features) hat allerdings kein solches Extremverhalten.

4.1.3 Änderungen von Anforderungen

Dieses Subkonzept ist fast skrupellos zu nennen. Der Vorteil davon ist, dass die Anforderungstreue auf diese Weise immer eine stimmige Momentaufnahme ist. Allerdings wird die Frage aufgeworfen, wieso Anforderungen die es mal in fixierter Form gegeben hat einfach verworfen werden, ohne eventuelle Entwicklungszeit zu würdigen. Sollte zum Zeitpunkt der Änderung einer Anforderung diese schon fertig implementiert sein, dann haben die Entwickler sich komplett an die Anforderungen des Kunden gehalten und gleichzeitig sinkt unter Umständen der Wert der Anforderungstreue.

Randfallbetrachtung

Im Extremfall würde sich eine Anforderung unmittelbar vor Ende des Projekts ändern, so dass sie nicht mehr zu implementieren ist und gleichzeitig die alte Funktion nicht

mehr gefordert wäre. Dieses hätte auf zwei Arten negative Auswirkungen auf die Anforderungstreue (entstandenes negatives Feature, schlecht bewertetes neues Feature). Dieses Verhalten ist extrem kundenfixiert und subjektiv fast als ungerecht für die Entwickler zu nennen. Dieses ist im Rahmen dieser Arbeit aber nicht als Nachteil zu betrachten. Die Anforderungstreue soll eine realistische Momentaufnahme der Kundenzufriedenheit sein, also ist dieses Verhalten sogar gewollt.

4.1.4 Nicht geforderte Funktionen

Das größte Problem dieses Konzepts ist es, dass der Kunde etwas eingeben muss, was ihn eventuell gar nicht so sehr interessiert. Bei umfangreicher Software kommt hinzu, dass es dem Kunden vielleicht gar nicht in kurzer Zeit möglich ist, alle Funktionen der Software zu überblicken. Zudem sind vielleicht einige auf den ersten Schein nicht geforderten Funktionen nötig, um andere Anforderungen des Kunden zu erfüllen.

Randfallbetrachtung

Sollte der Kunde sehr fein-granular bei der Erstellung von negativen Features vorgehen, so wird es davon im Extremfall eine große Anzahl davon geben. Existieren nun relativ wenig normale Features, dann hat dieses einen extrem negativen Einfluss auf die Anforderungstreue. Bei einem Szenario mit zehn angelegten negativen Features und einem MUST Feature kann die Anforderungstreue keinen Wert über 50% mehr erreichen, auch wenn das MUST Feature mit maximaler Zufriedenheit bewertet worden ist.

4.1.5 Bewertung

Das Bewertungssystem wurde von den Testpersonen (siehe oben) erwähnenswert positiv aufgenommen. Trotzdem mag es Probleme geben, die sich den Kunden gar nicht bewusst präsentieren. Wie in 'Five Stars Dominate Ratings' [You09] zu lesen ist (mit einer extrem hohen Anzahl an Testwerten), neigt diese Methode stark zu vollkommener Zustimmung oder Ablehnung (5 beziehungsweise 1 Stern(e)). Im Subkonzept wird dieser Nachteil abgeschwächt beziehungsweise vollkommen nichtig erklärt, mit der Begründung dass die Kunden ein hohes Maß an Motivation mitbringen ihre eingegebenen Features zu bewerten. Das schließt aber eine Beeinflussung des Konzepts auf bestimmte Werte (Richtung 1 oder 5 Sterne) nicht aus.
Es mag für Kunden auch schwierig sein abzugrenzen, ob ein Feature zum Beispiel 3 oder 4 Sterne verdient hat. Weiterhin mögen verschiedene Kunden das Konzept verschieden interpretieren. Einige mögen 5 Sterne erst bei absoluter Begeisterung vergeben und andere schon bei grundsätzlicher Zufriedenheit. Die Einstellung des Kunden hat somit auf jeden Fall Auswirkung auf die Anforderungstreue.

Randfallbetrachtung

Die Extremfälle dieses Subkonzept sind sehr robust. Prinzipiell werden mehrere Elemente mit einer Skala (1-5) bewertet und dann der durchschnittliche Wert berechnet. Dadurch gibt es keine Randfälle, die ein besonderes Verhalten an den Tag legen.

4.1.6 Berechnung der Anforderungstreue

Die Anforderungstreue wird wie in Formel 3.1 beschrieben berechnet. Bei diesem Verfahren gibt es einige Einflüsse seitens des Konzepts, auf die der Kunde keinerlei Einfluss hat. Die Relation zwischen den verschiedenen Prioritätsstufen (siehe oben) und den Einfluss von negativen Features. Der Kunde könnte seine Eingaben auf andere Ansichten gestützt haben und so ein Ergebnis vom Konzept geliefert bekommen, welches sich nicht mit seiner subjektiven Erfolgseinschätzung deckt.

Randfallbetrachtung

Da die Subtraktion aus Formel 2.1 in Formel 3.1 nicht übernommen worden ist, gibt es fragwürdiges Verhalten für extrem schlechte Projekte. Ein Projekt, welches keine Anforderungen des Kunden erfüllt wird immer eine Anforderungstreue von 0 vom Konzept erhalten. Was ist aber mit einem Konzept, welche zusätzlich noch nicht geforderte Anforderungen erfüllt, also komplett an den Kundenanforderungen vorbei? Diese Projekte sind eigentlich noch schlechter zu bewerten als die vorher erwähnten, erhalten aber von diesem Konzept trotzdem den gleichen Wert 0.
Diese Randfälle haben glücklicherweise keine dramatischen Auswirkungen, da nicht damit zu rechnen ist, dass es viele Projekte gibt, die so schlechte Ergebnisse erzielen. Das Konzept wurde optimiert für eine Vielzahl von 'normalen' (Anforderungstreue über 0) Projekten und liefert für diese Projekte Werte, mit denen sich die Kunden aus den Tests gut identifizieren konnten.

4.2 Grenzen des Konzepts

Der Umgang mit Anforderungen ist nicht einfach. Vor allem im Bereich der Softwareentwicklung. Das entwickelte Konzept kann kein menschliches Fingerspitzengefühl eines Requirements Engineer ersetzen. Es macht einige fixe Annahmen und ist bis zu einem gewissen Grad daher statisch, was bei zu erwartenden extrem unterschiedlichen Kunden zu Problemen führen könnte. Gerade die oben genannten Randfälle führen in der Regel zu ungenauen Werten der Anforderungstreue. Dieses Konzept ist daher nicht in der Lage für alle möglichen Projekte immer eine zutreffende Anforderungstreue zu berechnen.

4.3 Robustheit gegenüber nicht fundierten Eingaben

Da das Konzept einige eigene Annahmen trifft und Berechnungen ohne Kontrolle des Kunden durchführt, kann es zumindest teilweise nicht fundierte Eingaben ausgleichen. Sollte ein Kunde zum Beispiel nur COULD Features anlegen oder ähnliches Extremverhalten zeigen. Da der Kunde der einzige Akteur ist, bleibt die Bewertung zumindest innerhalb eines Projekts immer konsistent mit den Kundeneingaben. Gibt der Kunde also nur Dinge ein, die bei Erstellung des Konzeptes eigentlich anders erdacht worden sind, mag es trotzdem zu zufriedenstellenden Ergebnissen kommen – in den Augen des Kunden.

4.4 Der Kunde im Zentrum

Über das gesamte Konzept steht der Kunde im Mittelpunkt und hat die alleinige Kontrolle über alle Eingaben. Zumindest wenn es darum geht, wie mit Änderungen von Anforderungen umgegangen wird, scheint dies zumindest diskussionswürdig zu sein (siehe oben). Ist es richtig, dem Kunden die totale Kontrolle zu geben? Oder ist es erstrebenswert ein gewisses Machtgleichgewicht zwischen Kunden und Entwicklern herzustellen, um einen eventuell faireren Wert zu errechnen? Die Beantwortung dieser Fragen geht zumindest teilweise über dieses Konzept hinaus. Im Rahmen dieser Arbeit ist die Anforderungstreue eine kompromisslose Momentaufnahme der Zufriedenheit des Kunden und das ist in diesem Fall kein Nachteil. Kooperation zwischen Kunden und Entwicklern schließt dieses Konzept darüber hinaus nicht aus – dieses Problem geht aber weit über die Aufgabenstellung hinaus.
Ein jeder Kunde wird Interesse daran haben einen möglichst genauen Wert zu erreichen und die Macht, die er im Rahmen dieses Konzepts erhält, nicht absichtlich ausnutzen, um künstlichen Einfluss auf den Wert der Anforderungstreue zu nehmen. Und selbst wenn, so etwas kann nicht von einem statischen Konzept abgefangen werden.

4.5 Gesamtbetrachtung

Betrachtet man alle Subkonzepte dann stellt sich vielleicht an einigen Stellen die Frage 'ist das nicht zu einfach'? Tatsächlich wurden an vielen Stellen Kompromisse eingegangen, die zwar die Abläufe vereinfachen und so Kundenzeit sparen, aber unter Umständen negative Auswirkungen auf die Genauigkeit des Endergebnisses der Anforderungstreue haben können. Reichen drei Prioritätsstufen und fünf Zufriedenheitsstufen wirklich aus, um Basis einer Berechnung eines so schwierigen Themas wie Anforderungstreue zu sein? Die Antwort kann nicht definitiv sein, aber sie lautet: Das Konzept ist genau genug. Mit umfangreicheren Verfahren ist es mit Sicherheit möglich etwas genauere Ergebnisse zu erzielen, aber wie in Kapitel 3.4.2 bereits erwähnt wurde, waren alle Kunden mit den Ergebnissen zufrieden. Wie oben erwähnt ist das Ziel des Konzepts eine hinreichend genaue Tendenz der Anforderungstreue zu errechnen und

das Konzept anwendbar auf so viele Projekte wie möglich zu machen. Dieses Konzept auf ein beliebiges Projekt mit beliebigen Kunden anwenden zu können ist wichtiger, als die Genauigkeit der zweiten Nachkommastelle der Anforderungstreue.
Diese Betrachtungen führen zwangsläufig zur Frage: wie wichtig ist Genauigkeit? In der Informatik ist Genauigkeit nicht nur sehr wichtig, sondern oft essentiell und unabdingbar. Genau deshalb ist das Thema der Anforderungen in der Softwareumwelt so tiefgründig und interessant, denn hier liegt einer der natürlichen Konflikte. Anforderungen benötigen menschliche Interaktion und die Erfüllung der Anforderungen ist in Software sehr schwierig, unter Umständen gar nicht, zu messen. In dem entwickelten Konzept ist der Kunde, der kein in das Thema eingearbeiteter Experte ist, der einzige Akteur. Es wird daher mit einer gewissen Schwammigkeit zu rechnen sein und die sollten in jedem Falle kritisch hinterfragt werden – das Konzept ersetzt nicht das subjektive Erfolgsempfinden des Kunden.
Letztendlich ist ein Konzept entstanden, das nahezu auf beliebig geartete Projekte angewendet werden kann und die nötige zu investierende Kundenzeit auf ein Minimum reduziert, während es so genau bleibt, dass alle Kunden während der Tests sehr zufrieden mit den Ergebnissen waren. Man kann dieses Konzept also genau für den bestimmten Zweck verwenden.

5 Beschreibung des Prototypen

Im Rahmen dieser Arbeit wurde ein Prototyp entwickelt, um das Konzept in einem Webservice abzubilden und Tests damit durchzuführen. Dieser Prototyp wurde mit dem Play Framework 2.2.0 [Bor07] realisiert. Dieses ist ein Webanwendungs- Framework, welches die Programmiersprachen Java und Scala nutzt. In der Regel wird Java dabei für die innere Programmlogik benutzt, während Scala die Oberflächen (in diesem Fall HTML Seiten) dynamisch erweitern kann.
Das Programm ist strikt nach MVC Muster implementiert: Es existiert ein Datenmodell, welches alle nötigen Gegenstände es Konzepts abbildet. Weiterhin existieren sogenannte Controller, welche das Verhalten des Programms steuern und entscheiden, wie zum Beispiel auf Eingaben des Benutzers reagiert wird. Letztendlich gibt es die verschiedenen Ansichten, welche die Daten für den Kunden als HTML Seite darstellen. Abbildung 5.1 zeigt die Zusammenhänge zwischen den drei verschiedenen Klassenarten.

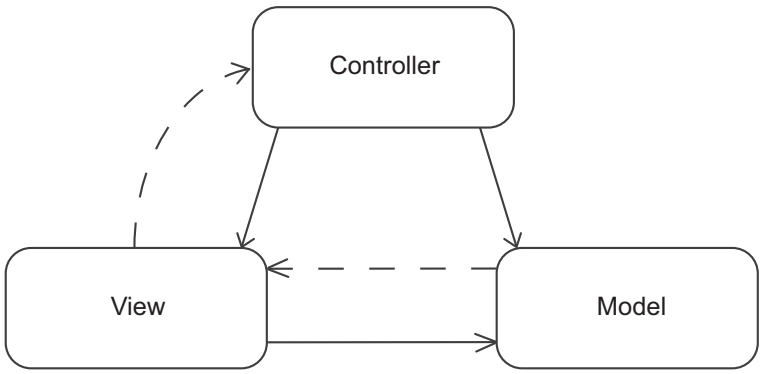

Abbildung 5.1: MVC Muster

5.1 Beschreibung der Klassen

Folgende Controller Klassen existieren:

- Application
  - Hauptprogrammlogik, welche entscheidet was bei einem Aufruf der Basis-URL geschieht.
- Features

  - Verwaltet das anlegen, editieren und löschen von Features.
- Projects
  - Verwaltet das anlegen, editieren und löschen von Projekten.
- Secured
  - Verwaltet die Berechtigungen der Benutzer.
- Users
  - Verwaltet das anlegen, editieren und löschen von Benutzern.

Im Webservice sind folgende Model Klassen implementiert:

- Feature
  - Die Modellklasse, die Anforderungen des Kunden in Form von Features darstellt.
- NegativeFeature
  - Diese Klasse bildet negative Features dar, welche benutzt werden um nicht geforderte Funktionen festzuhalten.
- Priority
  - Eine Hilfsklasse, um die Priorität eines Features festzuhalten.
- Project
  - Eine Modellklasse, welche ein Projekt darstellt.
- User
  - Eine Modellklasse, um Benutzer des Programms abzubilden.

Und schließlich sind sind folgende View Klassen implementiert:

- createfeature
  - Die Ansicht um Features zu erstellen.
- createnegativefeature
  - Die Ansicht um negative Features zu erstellen.
- editfeature
  - Die Ansicht um Features zu editieren.
- editnegativefeature
  - Die Ansicht um negative Features zu editieren.
- editproject
  - Die Ansicht um Projekte zu editieren.
- evaluateproject
  - Die Ansicht, auf welcher die Auswertung für vollendete Projekte angezeigt

wird.

- features
  - Die Ansicht von Features eines Projekts.
- index
  - Die Startseite für eingeloggte Benutzer: eine Liste aller Projekte.
- login
  - Eine Loginseite für nicht eingeloggte Benutzer.
- main
  - Das Haupttemplate, in welches alle anderen Ansichten eingefügt werden.
- negativefeature
  - Die Ansicht für ein negatives Feature.
- singlefeature
  - Die Ansicht für ein Feature.
- users
  - Eine Ansicht, um Benutzer anzulegen und zu löschen.

Abbildung 5.2 zeit ein UML 1.x Klasendiagramm der wichtigsten Modellklassen und ihrer Beziehungen.

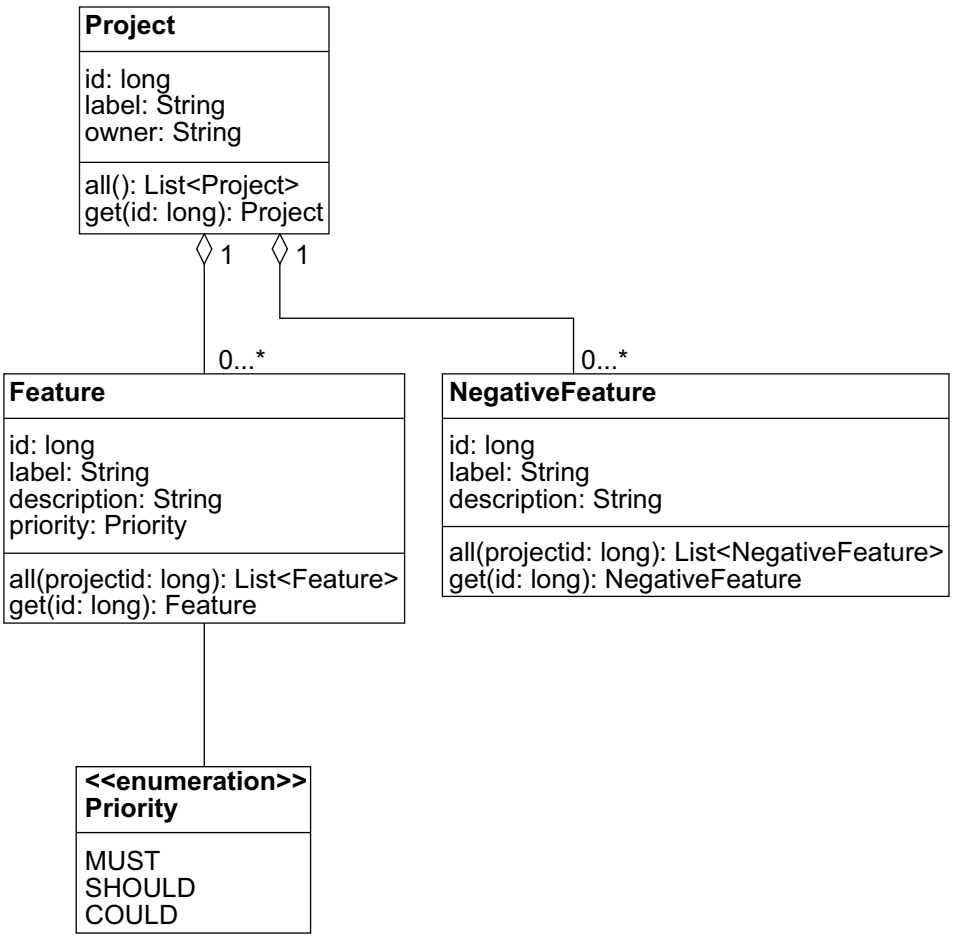

Abbildung 5.2: UML 1.x Klassendiagramm der wichtigsten Modellklassen

6 Fazit und Ausblick

In diesem Kapitel werden die Ergebnisse und Erkenntnisse der Arbeit zusammengefasst und ein Ausblick auf mögliche Vertiefungen in dieses Thema gegeben.

6.1 Fazit

In dieser Arbeit wurde ein Konzept zur Erfassung der Anforderungstreue in Softwareprojekten entwickelt. Mangels passender und konsistenter Definitionen wurde in diesem Rahmen eine Definition aufgestellt, welche sich auf die entwickelten Subkonzepte stützt.
Zunächst wurde der Begriff der Anforderung genauer erläutert, bevor anschließend die Anforderungstreue betrachtet wurde.
Das gesamte Verfahren wurde in 6 Teile getrennt, um schrittweise zu einem Ergebnis zu kommen.

1. Fixierung der Anforderungen
2. Priorisierung
3. Änderungen von Anforderungen
4. Bewertung
5. Nicht geforderte Funktionen
6. Berechnung der Anforderungstreue

Dabei ist jeder Schritt besonders darauf ausgerichtet möglichst einfach und schnell durchzuführen zu sein, denn die Akteure im Konzept werden Kunden sein, die nicht in ein festes Profil einzuordnen sind und deren Zeit knapp und kostbar sein wird.
Ein Webservice Prototyp wurde auf Basis des Play Framework entwickelt. Mit diesem Prototyp wurden Testläufe des Konzept mit (potentiellen) Kunden von Softwareprojekten durchgeführt. Die Ergebnisse der Tests wurden benutzt um Faktoren des Konzepts fein-abzustimmen.
Anschließend wurde sich kritisch mit dem entwickelten Konzept auseinander gesetzt. Dabei wurden die Schwachstellen jedes Subkonzepts betrachtet und besonderes Verhalten im Falle von Randfällen betrachtet.

6.2 Kritische Würdigung

Es wurde ein wissenschaftliches Konzept entwickelt, welches von Personen benutzt werden wird, die keinem einzugrenzendem Profil entsprechen und das in einem sehr schwierigen und konfliktreichen Bereich wie Softwareanforderungen. Da die Qualität der Ausgabe des Konzepts maßgeblich von den eingegebenen Daten abhängt, birgt dieses ein gewisses Risiko. Dem Kunden könnte es trotz Konzept schwer fallen, fundierte Daten einzugeben, die zu einem realistischen Ergebnis führen. Auf der anderen Seite wurden einige Subkonzepte so stark vereinfacht, dass dem Kunden möglichst viel Spielraum genommen wurde, damit er seine Eingaben schnell und unkompliziert machen kann. Ein Kunde, der gleichzeitig ein eingearbeiteter Experte im Thema Softwareanforderungen ist, könnte deshalb womöglich ohne das Konzept besser einschätzen, wie anforderungstreu ein Projekt ist.
Es ist auch unklar, wie genau die gelieferten Werte tatsächlich sind, da es keine Referenzdaten und -werte gibt, mit denen das Konzept getestet werden kann.

6.3 Ausblick

Ein wichtiger nächster Schritt wäre zunächst einmal das Konzept mittels fixer Referenzprojekte zu testen. Man könnte zum Beispiel die Anforderungstreue eines Projekts mit physischen Produkt (keine Software) mittels dem entwickelten Konzept bestimmen und dann die Ergebnisse mit den physischen Erfolgstests am Produkt vergleichen. Sollte das Konzept gute Ergebnisse im Bereich außerhalb von Software liefern, dann ist zu erwarten dass es dieses ebenfalls im schwierigen Softwareumfeld tun wird.
Erweiterungen des Konzepts wären ebenfalls möglich. Nimmt man Termine und Zeitpunkte von zum Beispiel Quality Gates auf, dann können die Ergebnisse dieser Daten ebenfalls mit in eine Endberechnung einfließen. Weiterhin wäre es möglich mehrere Messpunkte innerhalb eines Projekts festzuhalten, um einen zeitlichen Ablauf der Anforderungstreue zu messen. Dieses wäre insbesondere bei zum Beispiel agilen Projekten mit iterativen Entwicklungsphasen interessant. Auch Daten wie Budgeteinhaltung könnten aufgenommen werden, um das Konzept letztendlich zu einem Konzept zur Messung des gesamten Projekterfolges zu entwickeln.
Auch die Möglichkeit das Konzept schon während des Projekts anzuwenden und die vom Kunden eingegebenen Daten direkt an die Entwickler weiterzuleiten könnte erforscht werden. Es könnte damit eventuell positiver Einfluss auf den Projektablauf genommen werden und so eine höhere Anforderungstreue erreicht werden.
Eine Vertiefung des Themas in Richtung mobile Geräte wäre ebenfalls eine interessante Möglichkeit. Es könnte ein auf Android/iOS optimiertes Konzept samt Prototyp entwickelt werden.
Da die Anregung dieses Konzept zu benutzen in der Regel von Experten im Bereich Softwareanforderungen kommen wird und die Kunden dieses nur durchführen, wäre es ebenfalls interessant zu erforschen, ob neue Rollen das Konzept verbessern könnten. Vielleicht ist es möglich eine Art Requirements Engineer in das Konzept einzubinden, der den Kunden bei der Eingabe unterstützen kann – entweder um Zeit zu sparen oder

um genauere Ergebnisse zu erzielen.

So wie das agile Manifest empfiehlt keine Angst vor Änderungen an Anforderungen zu haben [Bec01] ist letztendlich ist zu empfehlen die gewisse Schwammigkeit im Umgang mit Softwareanforderungen nicht unbedingt immer als Nachteil zu empfinden, sondern als Chance – für Entwickler als auch deren Kunden – für enge Zusammenarbeit, die zu besserer Zufriedenheit führt.

Literaturverzeichnis

[BA04] K. Beck, C. Andres
Extreme Programming Explained (Second Edition)
Addison-Wesley Longman, 2004

[SLPK13] K. Schneider, O. Liskin, H. Paulsen, S. Kauffeld
Requirements Compliance as a Measure of Project Success
In Proceedings of Global Engineering Education Conference, IEEE, 2013

[CB04] D. Clegg, R. Barker
Case Method Fast-Track: A RAD Approach
Addison-Wesley, 2004

[Hei83] E. Heinen
Industriebetriebslehre
Betriebswirtschaftlicher Verlag Dr. Th. Gabler GmbH, 1983

[Dic51] H. F. Dickie
ABC Inventory Analysis Shoots for Dollars, not Pennies
In: Factory Management and Maintenance, 1951

[IEEE610] IEEE Standards Board
IEEE Standard Glossary of Software Engineering Terminology
IEEE Pres, 1990

[Dav58] M. Davis
Computability and Unsolvability
McGraw-Hill, 1958

[Wal01] E. Wallmüller
Software-qualitätsmanagement in der Praxis
Hanser Verlag, 2001

[KCHNP90] K. Kang, S. Cohen, J. Hess, W. Novak, A. Peterson
Feature-Oriented Domain Analysis (FODA) Feasibility Study
Software Engineering Institute Carnegie Mellon University, 1990

[KKLKKS98] K. Kang, S. Kim, J. Lee, K. Kim, G. Kim, E. Shin
FORM: A Feature-Oriented ReuseMethod with Domain-Specific Reference Architectures
Annals of Software Engineering, 1998

[CE00] K. Czarnecki and U. Eisenecker
Generative Programming: Methods, Tools, and Applications
Addison-Wesley, 2000

[Bos00] J. Bosch
Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach
ACM Press, Addison-Wesley, 2000

[CZZM05] K. Chen, W. Zhang, H. Zhao, and H. Mei
An Approach to Constructing Feature Models Based on Requirements Clustering
In Proceedings of the International Conference on Requirements Engineering, IEEE CS Press, 2005

[BSR04] D. Batory, J. Sarvela, and A. Rauschmayer
Scaling Step-Wise Refinement
IEEE Transactions on Software Engineering, 2004

[CHS08] A. Classen, P. Heymans, and P. Schobbens
What's in a Feature: A Requirements Engineering Perspective
In Proceedings of the International Conference on Fundamental Approaches to Software Engineering, Lecture Notes in Computer Science Band 4961, Springer-Verlag, 2008

[Zav03] P. Zave
An Experiment in Feature Engineering
In Programming Methodology, Springer-Verlag, 2003

[Bat05] D. Batory
Feature Models, Grammars, and Propositional Formulas
In Proceedings of the International Software Product Line Conference, Lecture Notes in Computer Science Band 3714, Springer-Verlag, 2005

[ALMK08] S. Apel, C. Lengauer, B. Möller, and C. Kästner
An Algebra for Features and Feature Composition
In Proceedings of the International Conference on Algebraic Methodology and Software Technology, Lecture Notes in Computer Science Band 5140, Springer-Verlag, 2008.

[Coh04] M. Cohn
User Stories Applied: For Agile Software Development
Addison-Wesley Professional, 2004

[Coc03] A. Cockburn
Use Cases effektiv erstellen
MITP Verlag, 2003

[You09] YouTube
Five Stars Dominate Ratings
Online, http://youtube-global.blogspot.de/2009/09/five-stars-dominate-ratings.html, 2009, abgerufen am 12.10.2013

[Bal08] H. Balzert
Lehrbuch der Softwaretechnik: Softwaremanagement

Spektrum Akademischer Verlag, 2008

[Rup09] C. Rupp
Requirements-Engineering und -management
Hanser Verlag, 2009

[PV06] D. Procaccino, J. M. Verner
Software project managers and project success: An exploratory study
In Journal of Systems and Software Band 79, 2006.

[Bor07] G. Bort (open source)
Play Framework
Online, http://www.playframework.com/, 2007, abgerufen am 25.07.2013

[Bec01] K. Beck und andere
Manifesto for Agile Software Development
Online, http://http://www.agilemanifesto.org/, 2001, abgerufen am 22.09.2013

Compact Disc

Der Inhalt der beiliegenden Compact Disc besteht aus:

Pfad	Beschreibung
/ausarbeitung/thesis_minks.pdf	Version dieser Ausarbeitung im PDF Format
/ausarbeitung/latex/	Version dieser Ausarbeitung im TEX Format
/prototyp/	Prototyp als Eclipse Projekt
/play-2.2.0/	Play Framework 2.2.0

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 25.12.2013

_____

Sebastian Minks