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

Entwicklung eines erfahrungsgestützten Editors für Qualitätsmodelle

Studienarbeit

im Studiengang Mathematik mit Studienrichtung Informatik

von

Stefan Keil

Prüfer: Prof. Dr. Kurt Schneider
Betreuer: M. Sc. Eric Knauss

Hannover, 19. Februar 2007

Danksagung

Ich bedanke mich bei Herrn M. Sc. Eric Knauss für die hervorragende Betreuung meiner Studienarbeit.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Qualitative Anforderungen, auch nicht-funktionale Anforderungen genannt, sind ein wichtiger Bestandteil während der Anforderungserhebungsphase in der Softwareentwicklung, der allerdings oft vernachlässigt wird.

Die Gründe hiervon reichen von einem zu knappen Zeitplan über mangelndes Problembewusstsein bis hin zu schlichtem Unvermögen. Qualität an einem immateriellen Produkt, wie Software, zu messen und zu prüfen, stellt eine Herausforderung dar. Eine ingenieurmäßige Fortschrittskontrolle und Qualitätsverbesserung lässt sich nur durch ein Qualitätsmodell realisieren, in dem Maßstäbe eingesetzt werden um Qualität präzise zu erfassen und auch wirklich ‚messen' zu können. So kann gezielt auf Qualitätsziele hingearbeitet werden. Damit ist außerdem eine einheitliche und übersichtliche Dokumentationsform gewählt, die später, z.B. in Reviews, als Referenz dienen kann.

Da es in der Software-Entwicklung auch um einen Erfahrungs-Sammlungs-Prozess geht, um Fehler zu vermeiden oder Zeit zu sparen, indem nicht alles doppelt gemacht wird, ist ein erfahrungsgestütztes Qualitätsmodell, das sowohl die vorhandene Erfahrung berücksichtigt und an den Anwender weitergibt, als auch um neue Erfahrungen erweiterbar ist, sehr wünschenswert.

1.2 Aufgabenstellung

In dieser Studienarbeit geht es um die „Entwicklung eines erfahrungsgestützten Editors für Qualitätsmodelle". Dafür soll ein bereits vorhandener erfahrungsbasierter Editor für funktionale Anforderungen erweitert werden, so dass er auch die Dokumentation nicht-funktionaler Anforderungen, also qualitativer Anforderungen, unterstützt und durch Kritiken die Erstellung aktiv leitet. Diese Kritiken werden in einer erweiterbaren Erfahrungsbasis gespeichert. Die Erweiterungen des Editors und der Erfahrungsbasis werden direkt am vorhandenen Quelltext durchgeführt.

Neben dem aktiven Unterstützen soll es die Möglichkeit geben, jede Kritik mit eigenen Erfahrungen zu kommentieren, und so noch zusätzlich gewonnene Informationen zu vermerken und für alle anderen Nutzer und weiteren Projekte verfügbar zu machen.

Die Dokumentation der einzelnen Qualitätsmerkmale soll mittels eines Qualitätsmodells und einem dafür entworfenen Formular stattfinden, dessen Eingabefelder bzw. Auswahlfelder durch Fehlermeldungen, Warnungen und Hinweise erläutert werden. Damit wird der Benutzer bei seinem Gedankengang, wie das Formular auszufüllen sei unterstützt, und es können häufig gemachte Fehler vermieden werden.

Außerdem können leicht zu jedem dieser Eingabe- und Auswahlfelder im Formular neue Hinweise in die Erfahrungsbasis eingefügt werden, die dann für alle Benutzer sichtbar sind. So kann auch während eines Projektes die gewonnene Erfahrung gesichert werden.

Das Erstellen neuer Fehler und Warnungen ist meistens nicht nur über die GUI des Editors möglich, sondern benötigt einen Eingriff in den Quelltext, oder zumindest die Kenntnis gewisser Java-Klassennamen. Dies zu beheben ist allerdings nicht Aufgabe dieser Studienarbeit.

Die Ausarbeitung einer guten Erfahrungsbasis für die Erstellung der Qualitätsanforderungen ist wünschenswert, jedoch liegt die primäre Aufgabe in den oben genannten Funktionen.

1.3 Der vorhandene Editor

Der vorhandene Editor wurde im Rahmen einer Master Arbeit[1] am Fachgebiet Software Engineering entwickelt. Mit ihm ist es möglich, Projekte anzulegen und diese zu speichern und zu laden. In den Projekten können Use Cases[2] angelegt, bearbeitet und gelöscht werden (sowie als HTML-Tabelle exportiert werden). Außerdem kann die Erfahrungsbasis bearbeitet, erweitert und kommentiert und das allgemeine Use Case Formular modifiziert werden.

Die GUI des Editors gliedert sich dabei in 5 Teile (Abbildung 6: Die Grafische Benutzeroberfläche des erweiterten Editors):

- *ProjectView:* Hier wird der so genannte ‚Projekt-Baum' angezeigt, in dem die Wurzel das Projekt selbst ist und an dieser alle Use Cases als Blätter anhängen. Diese Use Cases haben ihrerseits ihre Attribute als Blätter. Die Attribute werden mit einem Warnungs- oder Fehlersymbol versehen, wenn in der Erfahrungsbasis ein entsprechender Eintrag vorhanden ist. Hat mindestens eines der Attribute eine Warnung oder einen Fehler, so wird dieses Symbol auch bei dem Use Case selbst angezeigt und hat mindestens einer der Use Cases solch ein Symbol hat dies auch das Projekt selbst. (Wenn beides vorkommt wird nur ein Fehler angezeigt.)
- *ConstructionView:* Hier wird das Use Case Formular desjenigen Use Case angezeigt, der gerade bearbeitet wird. Dabei werden auch hier die jeweiligen Attribute mit den Warnungs- bzw. Fehlersymbolen versehen.
- *ProblemView:* Hier werden in einer Tabelle alle Warnungen und Fehler des gesamten Projektes angezeigt und können nach gewissen Kriterien sortiert werden (z.B. nach dem zugehörigen Use Case). Außerdem wird mit einem Doppelklick auf eine der Warnungen oder Fehler das zugehörige Use Case Formular an der relevanten Attributstelle in der ConstructionView angezeigt.
- *AssistentView:* Markiert man im Projekt-Baum oder in der ConstructionView ein Attribut eines Use Case, so werden hier die Warnungen und Fehler, sowie Hinweise zum Ausfüllen dieses Attributs beschrieben. Diese sind nach Relevanz sortiert und können nacheinander durchgeblättert werden. Sie können hier auch ausgeblendet werden, wenn man sich entschließt die Eingabe bei dem jeweiligen Attribut nicht zu ändern.
- *SimulationView:* Hier werden alle verlinkten Use Cases des jeweiligen Use Case in einem Diagramm angezeigt. Per Doppelklick kann man sich direkt das Formular eines dieser Use Cases in der ConstructionView anzeigen lassen.

1.4 Aufbau der Arbeit

Die Arbeit gliedert sich in drei Hauptabschnitte. Im ersten Abschnitt werden die Begriffe der Qualitätsanforderungen erläutert und der Zweck eines Qualitätsmodells verdeutlicht, sowie das verwendete Qualitätsmodell und das daraus abgeleitete benutzte Formular für den Editor beschrieben. Im zweiten Abschnitt werden die Anforderungen der zu entwickelnden Software dargelegt und ihre Priorisierung angegeben. Im dritten Abschnitt wird auf den Prozess der konkreten Software-Erweiterung des Editors eingegangen, der die größte Aufgabe dieser Studienarbeit darstellt und zeitlich den meisten Aufwand in Anspruch nahm. Anschließend werden noch Möglichkeiten, wie die Software weiter zu verbessern wäre, vorgestellt. Abschließend werden die erreichten mit den angestrebten Zielen verglichen und ein Fazit gezogen.

[1] [Cri06]
[2] [Coc01]

2 Qualitätsanforderungen

Qualität ist „die Beschaffenheit einer Einheit bezüglich ihrer Eignung, festgelegte und abgeleitete Erfordernisse (Qualitätsanforderungen) zu erfüllen.“[3]

Da zu Beginn ‚festgelegte Qualitätsanforderungen' bestimmt werden müssen, ist Qualität nur in Bezug auf Ziele definiert. Es gibt nicht *die* beste Qualität. Was für die eine Software hervorragend ist, kann für eine andere miserabel sein. Einige Qualitätsanforderungen können sich sogar widersprechen (z.B. Laufzeit- und Speichereffizienz). Es gilt also die Qualitätsziele zu erheben und anschließend zu priorisieren um zur *richtigen* Qualität einer Software zu gelangen.

2.1 Begriffsklärung

- Die *Qualitätsmerkmale* – auch Qualitätsaspekte genannt. – “stellen Eigenschaften einer Funktionseinheit dar, anhand derer ihre Qualität beschrieben und beurteilt wird.“[4] Dabei ist in dem Qualitätsmerkmal noch kein Grad über die Ausprägung enthalten. Mit ‚Funktionseinheit' ist im Rahmen dieser Studienarbeit ‚Software' gemeint.
- Das Maß für die Ausprägung des Qualitätsmerkmals wird durch einen *Qualitätsmaßstab* angegeben und der gewollte Grad der Ausprägung wird durch *Soll-Werte* und einer konkreten *Prüfungsdurchführung* beschrieben.
- Das *Qualitätsziel* ist das angestrebte Qualitätsmerkmal und die erreichte *Qualität* ist die Übereinstimmung des Ziels mit der Anforderung, die durch die angegebene Prüfung genau ermittelt werden kann.
- Die Qualitätsziele zusammen bilden die *Qualitätsanforderungen* des Projektes.
- Die Strukturierung der Qualitätsanforderungen bildet das *Qualitätsmodell*. Es ist ein Modell der verwendeten Qualitätsbegriffe.

2.2 Das verwendete Qualitätsmodell

Das Ziel eines Qualitätsmodells ist es, Qualität präzise fassen und messen zu können. Es dient der Fortschrittskontrolle, also in wie weit die Qualitätsanforderungen umgesetzt wurden, und der Qualitätsverbesserung, um abschätzen zu können, was sich durch gewisse Maßnahmen verbessern wird.[5]

Das verwendete Qualitätsmodell ist der Vorlesung „Software-Qualität“[6] entnommen und hat die Form eines azyklischen Graphen. Es beginnt mit allgemeinen Qualitätsmerkmalen, die schrittweise verfeinert werden, wird fortgesetzt mit Qualitätsmaßstäben, und endet mit den konkreten Soll-Werten und Prüfungsdurchführungen. Es gliedert sich somit in drei Schichten:

- Abstrakte Benennung des Qualitätsmerkmals:
  Hier werden die Qualitätsmerkmale in feste projekt-unabhängige Bereiche vorsortiert und man erhält als Software-Entwickler einen Überblick darüber, welche Bereiche es gibt. Dies dient als Checkliste um kein Qualitätsmerkmal zu vergessen und es bietet die Möglichkeit sofort eine grobe Priorisierung vorzunehmen.

[3] [Lig02] S. 5
[4] [Lig02] S. 5
[5] [Sch06] S. 66
[6] [Sch06]

- Konkrete Benennung des Qualitätsmerkmals und des Maßstabs:
  Hier findet die Übertragung des abstrakten Qualitätsmerkmals auf die individuelle Situation statt. Das konkrete Qualitätsmerkmal ist *der* projekt-spezifische Teil des Modells. Hier können keine Vorgaben gemacht werden, da Software zu verschieden ist. Für einige Maßstäbe gibt es häufig genutzte Metriken, auf die zurückgegriffen werden kann. Diese können in der Erfahrungsbasis gespeichert werden und so durch einen Hinweis an den Benutzer weitergegeben werden.
- Soll-Werte und Prüfungsdurchführungen:
  Hier wird angegeben, was genau die Prüfungsgegenstände sind und welcher Grad der Ausprägung des Qualitätsmerkmals gewünscht ist. Die Soll-Werte können später mit dem Ist-Zustand verglichen werden. In der Prüfungsdurchführung steht, wie der Ist-Zustand gemessen werden kann.

Über eine Einteilung der allgemeinen Qualitätsmerkmale gibt es in der Literatur viele verschiedene Angaben. So gibt Liggesmeyer[7] eine Auflistung von „Sicherheit, Zuverlässigkeit, Verfügbarkeit, Robustheit, Speicher- und Laufzeiteffizienz, Änderbarkeit, Portierbarkeit, Prüfbarkeit und Benutzbarkeit“[8] an. Andere benutzen eine hierarchische Struktur mit einer Verfeinerung der Merkmale, so z.B. Boehms ‚Qualitätenbaum’[9], der in einem dreistufigen Baum, die allgemeinen Qualitätsmerkmale entwickelt. In der ISO/IEC 9126[10] wird ein zweistufiger Baum verwendet, bei dem die Produktqualität mit sechs allgemeinen Qualitätsmerkmalbereichen ausgedrückt werden kann, die sich dann in Teilmerkmale gliedern. Aufgrund seiner Übersichtlichkeit wurde dieser Qualitätsmerkmal-Baum gewählt und soll in dem Formular verwendet werden.

- *Funktionalität*:
  Angemessenheit, Richtigkeit, Interoperabilität, Sicherheit und Konformität
- *Zuverlässigkeit*:
  Reife, Fehlertoleranz, Robustheit, Wiederherstellbarkeit und Konformität
- *Benutzbarkeit*:
  Verständlichkeit, Erlernbarkeit, Bedienbarkeit, Attraktivität und Konformität
- *Effizienz*:
  Zeitverhalten, Verbrauchsverhalten und Konformität
- *Änderbarkeit*:
  Analysierbarkeit, Modifizierbarkeit, Stabilität und Prüfbarkeit
- *Übertragbarkeit*:
  Anpassbarkeit, Installierbarkeit, Koexistenz, Austauschbarkeit und Konformität

Das Qualitätsmodell sieht außerdem vor, dass ein konkretes Qualitätsmerkmal durchaus mit mehreren Qualitätsmaßstäben gemessen wird, die wiederum mehrere Soll-Werte haben können mit jeweils einer eigenen Prüfungsdurchführung. Dieser Abschnitt des Modells ist also nicht statisch, da er sich den unterschiedlichen Software-Projekten anpassen können muss.

Der Einsatz des Qualitätsmodells geschieht dann in folgenden Schritten: Zuerst werden die angestrebten Qualitätsmerkmale ermittelt und priorisiert. Dann werden Prüfungen und Testfälle erstellt und gezielt auf die Qualitätsziele hingearbeitet. Anschließend kann eine Bewertung der erreichten Ziele und eine Interpretation der Testergebnisse erfolgen.[11]

---

[7] [Lig02]
[8] [Lig02] S. 6
[9] [Boe81]
[10] [Wik07]
[11] [Sch06] S. 70

2.3 Das verwendete Formular

Das Formular soll eine schriftliche Umsetzung des Qualitätsmodells sein und somit eine Dokumentationsform für eine Qualitätsanforderung darstellen. Dafür wurde das dreischichtige Qualitätsmodell in zwei Teile geteilt. Die erste Schicht und das konkrete Qualitätsmerkmal aus der zweiten Schicht gehören zum ersten Teil. Der Qualitätsmaßstab aus der zweiten Schicht und die dritte Schicht gehören zum zweiten Teil.

Der erste Teil bildet den Kopf des Formulars. Durch zwei hintereinander gesetzte Auswahl-Menüs kann das allgemeine Qualitätsmerkmal gewählt werden, indem zuerst eines der oben genannten sechs Qualitätsmerkmalbereiche und dann dazu entsprechend eines der Teilmerkmale ausgewählt wird. Durch die Angabe des konkreten Merkmals wird noch eine Namensgebung erzwungen. Ohne diese Angaben kann die Qualitätsanforderung nicht sinnvoll im Projekt-Baum angezeigt werden. Sie stellen Pflichteingaben dar. Es ist auch nötig mit diesen Angaben zu beginnen, da man sich bei der Erhebung der Qualitätsanforderungen einer Software zuerst Gedanken darüber machen muss, von welchen Eigenschaften der Software man die Qualität beschreiben oder beurteilen möchte. Das Formular fördert dadurch diese Gedankenreihenfolge.

Danach wird in das Formular ein Info-Bereich eingeschoben, der dem zweiten Teil des Qualitätsmodells vorangestellt wird. Dieser Info-Bereich gehört nicht mit zum Qualitätsmodell, sondern ist ihm übergeordnet und stellt projekt- und unternehmensbezogene Daten dar, also Angaben, die nicht die Qualität der Software beschreiben, sondern für die Anforderungserhebungsphase und allgemeiner Organisation von Interesse sind. Dies können z.B. der Ersteller dieser Qualitätsanforderung sein, oder eine unternehmensinterne ID, die Priorisierung der Anforderung oder der Status der Ausgereiftheit der Dokumentation (Entwurf, Reviewed, etc.). Um Qualitätsmerkmale nach Belieben zusammenfassen zu können, wurde der Punkt ‚Qualitätsgruppe' eingefügt, bei dem neue Gruppennamen erstellt werden können und dann ein Qualitätsmerkmal einer dieser Gruppen zugewiesen werden kann. Der Info-Bereich soll später vom Benutzer frei modifizierbar sein, weil verschiedene Benutzerkreise möglicherweise einen unterschiedlichen Bedarf an Info-Punkten haben (siehe 4.4, Qualitätsanforderungen: 1. Punkt).

Die ersten beiden Bereiche des Formulars sollen vollständig auf einem Bildschirm sichtbar sein bzw. auf eine Seite passen, um schnell und gut überblickt werden zu können. Den Abschluss des Formulars bildet der zweite Teil des Qualitätsmodells. Hier können beliebig viele Qualitätsmaßstäbe und zu jedem Qualitätsmaßstab beliebig viele Soll-Werte mit Prüfungsdurchführungen hinzugefügt werden. Dadurch kann dieser Teil sehr lang werden. Da alle anderen Daten bereits auf der ersten Seite des Formulars sichtbar sind, sind auch lange Qualitätsanforderungen übersichtlich und gut strukturiert.

3 Anforderungen (der zu erstellenden Software)

Die Anforderungen an eine Software werden vom Kunden gestellt und dienen sowohl ihm selbst als auch den Programmierern dazu, zu überprüfen, ob die Software das leistet, wozu sie entwickelt wurde. Die Zielbestimmungen und funktionalen und qualitativen Anforderungen der zu erstellenden Software sind ähnlich mit denen der bestehenden Software, so dass sie nur kurz aufgegriffen werden. Die Benutzergruppe bleibt identisch. Unabhängig von dieser Arbeit wurden die Systemvorrausetzungen von Java 1.5 auf Java 1.6 angehoben.

3.1 Zielbestimmung

Mit dem Editor sollen zusätzlich noch die Qualitätsanforderungen eines Projektes erstellt, gelöscht, bearbeitet und exportiert werden können. Das Projekt soll dabei als Ganzes verwaltet werden können. Es soll jederzeit zwischen dem Erstellen und Bearbeiten von Use Cases oder Qualitätsanforderungen gewechselt werden können. Der Benutzer soll immer den vollen Überblick über das gesamte Projekt haben und sich die mit einem Fehler oder einer Warnung versehenen Stellen der Use Cases oder Qualitätsanforderungen anzeigen lassen können. Das Konzept der aktiven Unterstützung durch Fehler, Warnungen und Hinweise, die nach Relevanz sortiert sind, wird beibehalten und die Erfahrungsbasis um entsprechende Meldungen erweitert.

3.2 Funktionale Anforderungen

Der Benutzer soll weiterhin Projekte neu erstellen, speichern und laden können, und alle Funktionen des Editors bezüglich der Use Cases sollen erhalten bleiben. Die komponentenweise Aufteilung[12] der Funktionen des bestehenden Editors wurde übernommen und lediglich um die Anordnungskomponente erweitert, die es vorher auch schon gab, nur nicht explizit erwähnt wurde. In ihr kann der Nutzer angeben, nach welchen Kriterien er die Projektanforderungen sortieren möchte.

Neu hinzu kommen folgende funktionale Anforderungen:

- *Projektverwaltungskomponente*:
  Der Benutzer kann Qualitätsanforderungen erstellen und löschen. Es soll, wie für die Use Cases, eine Export-Funktion als HTML-Tabelle der Qualitätsanforderungen geben.
- *Konstruktionskomponente*:
  Der Benutzer kann Qualitätsanforderungen bearbeiten. Dabei können auch beliebig viele Qualitätsmaßstäbe und für jeden Qualitätsmaßstab beliebig viele Soll-Werte hinzugefügt (und auch wieder entfernt) werden. Der Benutzer kann außerdem von ihm beliebig bezeichnete Qualitätsgruppen hinzufügen und wieder entfernen.
- *Argumentationskomponente* (keine neuen Funktionen):
  Der Benutzer kann die Erfahrungsbasis auch um Fehler, Warnungen und Hinweise für Qualitätsanforderungen erweitern, bestehende Kritiken bearbeiten und wieder löschen (in gleicher Manier, wie für die Use Cases, deswegen müssen die bestehen Funktionen nur abgeändert werden und es kommen keine neuen dazu).

[12] [Cri06] S.21ff.

- *Spezifikationskomponente* (keine neuen Funktionen):
  Hier gibt es keine neuen Funktionen, da der Benutzer das Qualitätsanforderungsformular, genau genommen das Qualitätsmodell, nicht bearbeiten können soll (siehe 4.4, Qualitätsanforderungen: Punkt 1).
- *Anordnungskomponente*:
  Der Benutzer kann die Qualitätsanforderungen nach abstrakten Qualitätsmerkmalen und nach der Qualitätsgruppe sortieren.

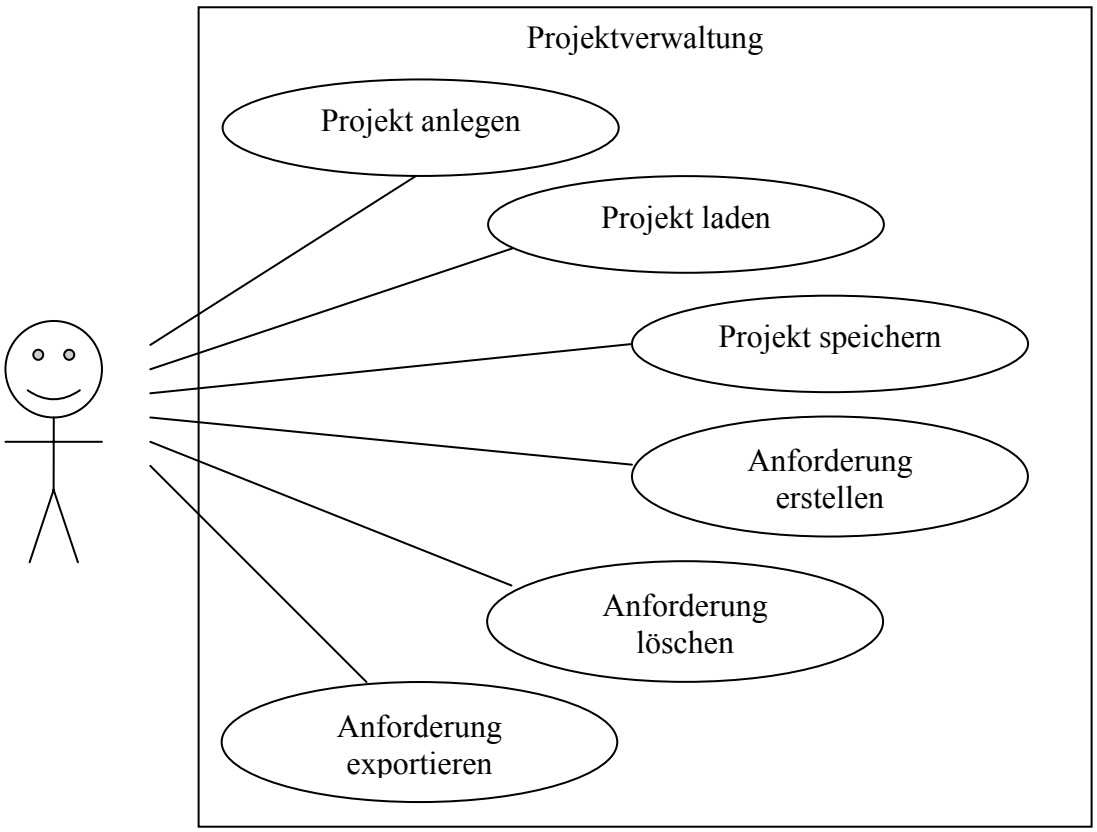

Abbildung 1: Use Case Diagramm: Projektverwaltung

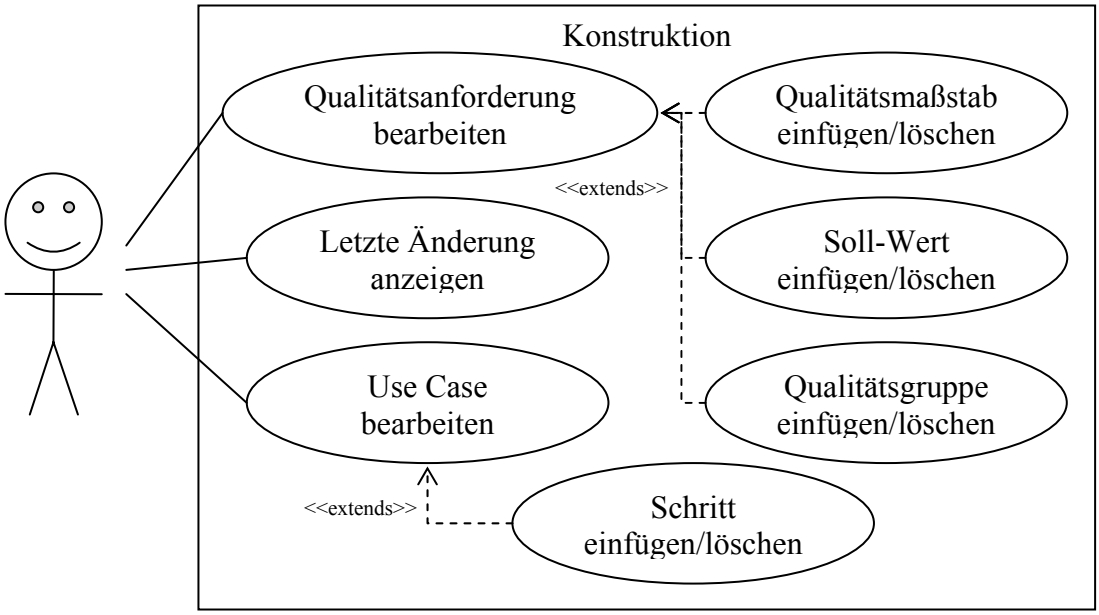

Abbildung 2: Use Case Diagramm: Konstruktion

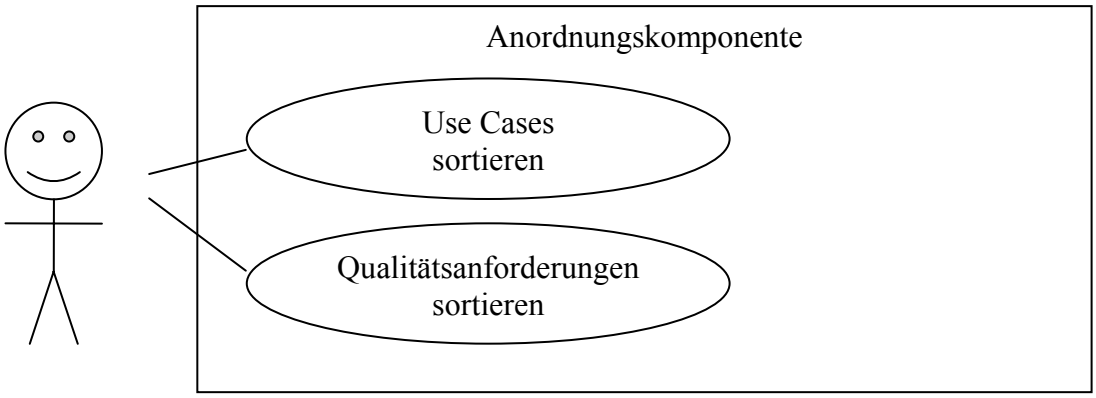

Abbildung 3: Use Case Diagramm: Anordnen

Die Use Cases für die neuen funktionalen Anforderungen wurden mit dem bestehenden Editor geschrieben. Hier ein Bespiel (weitere auf der CD):

Use Case ID:	21		
Titel:	Bearbeiten einer Qualitätsanforderung		
Systemgrenzen (Scope):	Konstruktion		
Ebene:	Benutzerebene		
Vorbedingung:	Die Qualitätsanforderung existiert bereits.		
Mindestgarantie:	Andere Projektdaten und nicht veränderte Daten bleiben erhalten.		
Erfolgsgarantie:	Die gemachten Änderungen an der Qualitätsanforderung sind übernommen.		
Stakeholder:	Benutzer:	will Qualitätsmaßstäbe erstellen bzw. löschen.	
	Konstruktion:	Datenkonsistenz	
Hauptakteur:	Benutzer		
Auslöser:	Benutzer möchte ein Qualitätsanforderung ändern, bzw. ergänzen.		
Hauptszenario:	1	Benutzer	wählt Qualitätsanforderung aus.
	2	Konstruktion	zeigt Qualitätsanforderung an.
	3	Benutzer	ändert Daten der Qualitätsanforderung.
	4	Konstruktion	übernimmt Daten.
	5	Kritiksystem	validiert die Daten und aktualisiert die aktuellen Kritiken.
Erweiterungen:	*3a: Benutzer erstellt oder löscht einen Qualitätsmaßstab:*		
	1	Benutzer	210: Hinzufügen oder Löschen eines neuen Qualitätsmaßstabs
	3b: Benutzer erstellt oder löscht einen Soll-Wert:		
	1	Benutzer	211: Hinzufügen oder Löschen eines neuen Soll-Wertes
	3c: Benutzer erstellt oder löscht eine Qualitätsgruppe:		
	1	Benutzer	212: Hinzufügen oder Löschen einer neuen Qualitätsgruppe
Priorität:	wichtig		
Performance:	Minuten bis eine Stunde (Routiniertheit des Benutzers)		
Verwendungshäufigkeit:	regelmäßig		

Abbildung 4: Use Case 21: Bearbeiten einer Qualitätsanforderung

3.3 Qualitative Anforderungen

Die qualitativen Anforderungen des zu erstellenden Editors decken sich mit denen des vorhandenen Editors. Die Priorisierung der abstrakten Qualitätsmerkmale wurde also schon vorgenommen.[13]

Der wichtigste Bereich ‚Benutzbarkeit' wurde so beschrieben, dass der Editor „unbedingt selbsterklärend" sein soll und der Benutzer ihn „auf Anhieb ohne große Einführung benutzen" können soll. Außerdem soll der Editor dem Benutzer „einen Vorteil" bringen, also „nicht zusätzlich Arbeit hervorrufen", sondern der Benutzer soll mit dem Editor „effektiv arbeiten" und „Zeit einsparen" können.[14] Alle diese Forderungen gelten auch für den erweiterten Editor, also für das Arbeiten mit den Qualitätsanforderungen, sowie für das Verwalten des gesamten Projektes, also die gleichzeitige Übersicht über die Use Cases und die Qualitätsanforderungen und die Bearbeitung der Erfahrungsbasis für beide Anforderungstypen. Hauptsächlich hier kommen neue Qualitätsanforderungen hinzu, bzw. bestehende müssen angepasst werden. Diese Qualitätsanforderungen wurden unter der Qualitätsgruppe ‚Intuitive Bedienung' zusammengefasst und dann für die oben erwähnten vagen qualitativen Anforderungen konkrete Qualitätsmerkmale und –maßstäbe herausgearbeitet. Mit dem fertigen Editor wurden die Anforderungen nachträglich noch einmal dokumentiert.

Hier ein Beispiel (weitere auf der CD):

Allg. Q-Merkmal:	Benutzbarkeit	
	Erlernbarkeit	
Konkretes Q-Merkmal:	Erlernbarkeit der Benutzeroberfläche	
	Q-Merkmal ID:	1
	Qualitätsgruppe:	Intuitive Bedienung
	Priorität:	wichtig
	Status:	Entwurf
Q-Maßstäbe:	Q-Maßstäbe:	Die Baumstruktur des Projektes
	Soll-Wert:	Das Projekt bildet die Wurzel des Baumes mit zwei festen Blättern "Use Cases" und "Qualitätsanforderungen", die immer sichtbar sind und mit dem gängigen Ordnersymbol aus Betriebssystemen versehen sind. An diese beiden Blätter werden dann wieder als Blätter die neu erstellten Use Cases und Qualitätsanforderungen gehängt. Klickt man auf eines dieser Blätter, so wird diese Anforderung angezeigt.
	Prüfung:	Benutzer soll zwei Anforderungen erstellen und hat dann 10 Sekunden Zeit, um die erste wieder anzeigen zu lassen.
	Soll-Wert:	Der Baum erfüllt Eigenschaften, wie sie in üblichen Dateimanagern oder in Eclipse vorkommen, wie: * Menü auf rechter Maustaste (mit u. a. Sortier- und Lösch-Funktion) * Mit Doppelklick Blätter eines Knoten zeigen bzw. verbergen * Die markierte Stelle ist farblich hinterlegt
	Prüfung:	Benutzer soll zwei Anforderungen erstellen, dann hat er 60 Sekunden Zeit um ein bestimmtes Attribut auszuwählen, danach die Sortiermöglichkeit zu zeigen und dann eine der Anforderungen wieder zu löschen.

[13] [Cri06] S.24 (Tabelle)
[14] [Cri06] S.24f.

	Q-Maßstäbe:	Die Buttons im Formular
	Soll-Wert:	Erklärender Roll-Over-Text
	Prüfung:	Geht man mit der Maus auf einen Button und drückt keine Maustaste und bewegt die Maus nicht mehr so kommt nach spätestens 2sec eine Textfeld, in dem eine kurze Beschreibung der Funktion dieses Buttons steht.

Abbildung 5: Qualitätsmerkmal: Benutzeroberfläche

3.4 Priorisierung der Ziele

Das oberste Ziel ist ein ausführbares Programm, das nicht abstürzt (z.B. durch Exceptions) oder womöglich gar nicht startbar ist. Außerdem sollen alle Funktionen bezüglich der Use Cases erhalten bleiben.

Solch ein Programm lag bereits vor, so dass an die Stelle des ersten Ziels, die Aufgabe eine neue Abstraktionsschicht einzuführen, getreten ist. Diese Schicht soll allgemein mit Projektanforderungen arbeiten können, unabhängig davon, ob es ein Use Case oder eine Qualitätsanforderung ist. Der vorhandene Quelltext musste darauf geprüft werden, ob er Use-Case-spezifisch ist oder allgemein für Anforderungen gelten soll und dann entsprechend geändert werden (siehe 4.2).

Weiterhin hohe Priorität haben das neue Formular und die Funktionen für das Erstellen, Löschen und Bearbeiten von Qualitätsanforderungen. Mittlere Priorität haben das Exportieren von Use Cases und die Sortiermöglichkeiten im Projektbaum, so dass dort vorerst nur eine Standard-Möglichkeit implementiert wird.

Geringe Priorität hat die Erstellung einer Erfahrungsbasis für die Qualitätsanforderungen. Dieser Punkt wurde bewusst an das Ende der Liste gehängt, da im ersten Schritt, beim Einführen der neuen Abstraktionsschicht, auch die Erfahrungsbasis verallgemeinert wurde, so dass es jedem Benutzer des Programms möglich ist, selber eine Erfahrungsbasis zu erstellen oder die vorhandene zu erweitern.

4 Erweiterung des bestehenden Editors

Die Erweiterung des bestehenden Editors und der damit einhergehenden Einarbeitung in einen fremden Quelltext stellte sich als zeitlich sehr anspruchsvoll heraus und nahm den größten Teil dieser Studienarbeit in Anspruch.

Der Java-Code war nahezu vollständig unkommentiert (nicht einmal der Klassenname war mit einer kurzen Javadoc Zeile versehen), so dass der gesamte Code wirklich ‚gelesen' werden musste, um ihn verstehen und sich einarbeiten zu können, um eine sinnvolle Verallgemeinerung des Codes auf Use Cases und Qualitätsanforderungen vornehmen zu können.

4.1 Erweiterung der GUI

Die erste Frage bei der Erweiterung des vorhandenen Editors war, wie die neuen Funktionen für den Benutzer erreichbar sind und angezeigt werden sollen. Aufgrund der ähnlichen Funktion ein Qualitätsanforderungsformular auszufüllen im Vergleich zu einem Use Case Formular, erschien eine grundsätzliche Beibehaltung der Fensteraufteilung als sinnvoll (siehe 1.3).

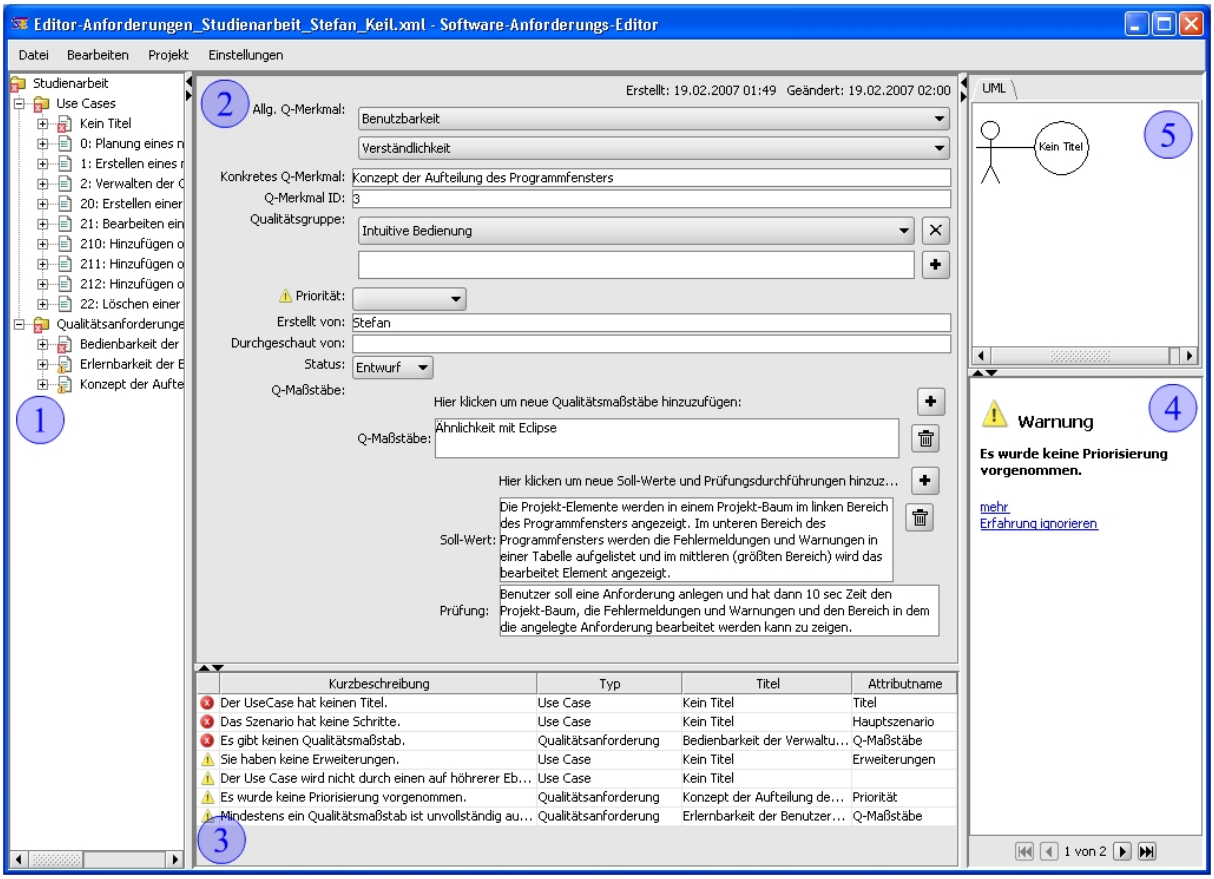

Abbildung 6: Die Grafische Benutzeroberfläche des erweiterten Editors

1) *ProjektView* mit Projekt-Baum, 2) *ConstructionView* mit Qualitätsanforderungs-formular, 3) *ProblemView* mit Auflistung aller Fehler und Warnungen, 4) *AssistentView* mit den Kritiken zu dem gewählten Feld im Formular, 5) *SimulationView* mit Diagramm eines Use Case (keine Verwendung bei Qualitätsanforderungen)

Für die *ProjectView* wurde auf Grund der Benutzbarkeit eine gemeinsame Anzeige gewählt, die beide Anforderungstypen gleichzeitig anzeigt und das Projekt mittels eines Projekt-Baumes darstellt, in dem alle funktionalen und qualitativen Anforderungen enthalten sind. Die Anzeige der Warnungs- und Fehlersymbole wurde beibehalten.

In der *ConstructionView* wurden zwei unterschiedliche Anzeigen gewählt, die sich aber im Aufbau ähneln (links Eigenschaftsname, daneben Eingabefeld und ganz rechts Buttons für Erweiterungsmöglichkeiten) und zwischen denen automatisch gewechselt wird, je nachdem welcher Anforderungstyp angezeigt wird.

Die Fehlermarkierung und Einbindung der Hilfe blieben gleich, so dass in der *AssistentView* gar nichts und in der *ProblemView* lediglich die Tabelle um die Spalte ‚Typ' erweitert werden musste.

Die *SimulationView* wird für die Qualitätsanforderungen nicht genutzt.

4.2 Erweiterung und Refactoring des Quelltextes

Der größte Arbeitsaufwand für diese Studienarbeit bestand darin (neben dem Lesen und Verstehen), den Quelltext lesbarer und erweiterbarer zu machen. Dies sind Methoden des Refactoring[15]. Dabei wird die Funktionalität des Quelltextes nicht verändert, sondern es findet ausschließlich eine Strukturverbesserung statt. Das Ziel war das Einführen einer neuen Abstraktionsschicht. Diese Schicht soll nur noch mit Projekt-Anforderungen und nicht mit Use Cases oder Qualitätsanforderungen arbeiten. Vom Datentyp des Use Case sollte abstrahiert werden und ein allgemeiner Datentyp eingeführt werden, der einen Use Case oder eine Qualitätsanforderung repräsentieren kann. Viele Datenstrukturen mussten daher durch eine verallgemeinerte Datenstruktur ersetzt werden. Um eine sinnvolle Namensgebung beizubehalten, musste in großem Umfang eine Umbenennung von Variablen, Methoden, Klassen und Packages stattfinden. Einige Packages wurden in einen allgemeinen und einen spezifischen Teil zerlegt. So sollte der Quelltext insgesamt leichter funktional erweiterbar sein.

Während dieser Aufarbeitung des Quelltextes fiel auf, dass es zwar ein MVC-Pattern für die Projektdaten gab, die aktuelle Situation, was wo gerade angezeigt werden soll und was markiert ist, davon jedoch entkoppelt war. Diese Daten existierten nur in den Views selbst und wurden untereinander von diesen ausgetauscht und gespeichert. Dieses wurde geändert und das vorhandene MVC-Pattern erweitert. Dabei wurde auch das Verhalten der Observer und Observables überarbeitet.

Für die Erstellung der neuen Kritiken wurde eine parallele Erweiterung und keine Verallgemeinerung gewählt, was noch getan werden könnte (siehe 4.4, Erfahrungsbasis verwalten:). Diese Erweiterung wird voraussichtlich als eigene studentische Arbeit vergeben werden.

Letztendlich musste die grafische Benutzeroberfläche des Editors noch erweitert werden, damit auch die neuen Funktionen des Editors angezeigt werden können. Außerdem wurden viele Quelltext-Teile kommentiert und mit Javadoc versehen.

4.3 Erweiterung der Erfahrungsbasis

Es wurden Prüfmethoden für Fehlermeldungen über ungenügend ausgefüllte Qualitätsanforderungen erstellt. Diese zeigen das Fehlen des allgemeinen und konkreten Qualitätsmerkmals, sowie mindestens eines angegebenen Maßstabs mit Soll-Werten und Prüfungsdurchführungen, an und melden doppelt vergebene IDs. Warnungen weisen auf eine fehlende Priorisierung und unvollständig ausgefüllte Qualitätsmaßstäbe hin. Das Hinzufügen, Bearbeiten und Löschen von Hinweisen für jedes Eingabe- und Auswahlfeld des

[15] [Fow99]

Qualitätsanforderungsformulars ist vollständig möglich, so dass dies von jedem Benutzer vorgenommen und dadurch im Laufe der Zeit die Erfahrungsbasis erweitert werden kann. Zu jedem der Eingabe- und Auswahlfelder wurden auch schon Hinweise hinzugefügt, die Tipps zum sinnvollen Ausfüllen dieses Feldes geben.

4.4 Weitere Verbesserungsmöglichkeiten

Sowohl im Laufe der Erstellung des erweiterten Editors als auch während des Arbeitens mit diesem, sind mehrere Verbesserungsmöglichkeiten aufgefallen, die hauptsächlich qualitativer Natur sind und dem abstrakten Qualitätsmerkmal ‚Benutzbarkeit → Bedienbarkeit' angehören. Einige Beispiele:

1. ProjectView:
   - Den Projekt-Baum modifizierbar machen, also vom Benutzer Unterordner einfügen lassen, damit dieser individuell die Anforderungen sortieren kann. (Dient der Übersicht bei großen Projekten.)
2. AssistentView:
   - Bei den Erläuterungen der Kritiken einen Button einfügen, der direkt die Editierfunktion dieser Kritik in der Erfahrungsbasis aufruft. (Spart den Umweg, erst die Erfahrungsbasis aufrufen und dann in einer Tabelle die angezeigte Kritik auswählen zu müssen.)
3. ProblemView:
   - Alternative Ansichten anbieten (z.B. über Reiterkarten wählbar), bei denen nur die Fehler und Warnungen der Use Cases oder der Qualitätsanforderungen stehen. (Dient der Übersicht bei großen Projekten und spart dem Benutzer Zeit beim Finden eines bestimmten Fehlers.)
4. ConstructionView:
   - Mittels Reiterkarten zwischen den zuletzt geöffneten Anforderungen schnell hin und her wechseln können (z.B. wie bei Eclipse mit den Klassen). (Spart Zeit bei großen Projekten im Projekt-Baum die jeweilige Anforderung zu suchen.)
   - Wenn man mehr Text schreibt als in ein Tabellenfeld passt, dann Tabellenfeld um eine weitere Zeile erweitern, damit der gesamte Text angezeigt werden kann. (Spart Zeit beim Lesen und weiteren Ausfüllen der Anforderung, da man nicht extra in die Tabellenzeile hineingehen muss um im Text zu scrollen.)
   - Wenn man bei einem Push-Down-Menü etwas ausgewählt hat, ist der gewählte Punkt nicht gespeichert bis man bei der gleichen Anforderung ein anderes Feld im Formular markiert. Die gewählte Information geht verloren, wenn man direkt (z.B. im Projekt-Baum) eine andere Anforderung auswählt. (Dies ist ein unerwartetes Verhalten des Editors, da ein Push-Down-Menü ein in vielen Anwendungen bekanntes GUI-Element ist und sich sonst nicht so verhält. Der Benutzer würde dies als funktionalen Fehler des Editors werten.)
5. Export für Qualitätsanforderungen:
   - Im Auswahlfenster, in dem die Qualitätsanforderungen gewählt werden, die exportiert werden sollen, diese nach Kriterien (Status, Priorität, abstraktes Qualitätsmerkmal, Qualitätsgruppe) sortieren können. (Spart Zeit beim Finden der gewünschten Qualitätsanforderungen.)
   - Einstellen können, ob die Qualitätsanforderungen in einzelnen Tabellen ausgegeben werden sollen, oder hierarchisch in großen Tabellen (sortiert nach dem allgemeinen abstrakten Qualitätsmerkmal).
6. Allgemein:
   - Mehrere Projekte gleichzeitig geöffnet haben können (wie z.B. in MS Word oder Adobe Acrobat Reader). (Spart Zeit, wenn man schnell etwas in einem anderen Projekt nachsehen möchte, wenn dort z.B. ähnliche Anforderungen sind.)

- Anforderungen kopieren und einfügen können. (Spart Zeit, wenn man in einem Projekt ähnliche Anforderungen hat oder wenn man in einem neuen Projekt eine neue Anforderung erstellen möchte, die ähnlich zu einer Anforderung aus einem alten Projekt ist.) Diese Funktion wurde in der Master Arbeit gewollt ausgeschlossen, da bezweifelt wurde, dass die so entstehenden Use Cases fehlerfrei seien, da der Benutzer seiner Arbeit vertraue und sie nicht mehr genügend kontrolliere.[16] Der Nutzen ist allerdings größer als das Risiko, denn erstens sollte eine Anforderung von einer zweiten Person durchgesehen werden, zweitens sollte sich auch jeder, der sich mit Softwareanforderungen beschäftigt hat, dieser Gefahr bewusst sein, drittens geht es bei der Software Engineering darum, Vorhandenes wieder zu benutzen und viertens ist diese Funktion von sehr großem Nutzen, wenn mehrere Personen mit diesem Editor (Teil-) Projekte ausgearbeitet haben, die dann zu einem großen Projekt zusammengefügt werden sollen.

7. Qualitätsanforderungen:
   - Den Info-Bereich des Formulars vom Benutzer modifizieren lassen können, so dass neue Punkte hinzugefügt und wieder entfernt werden können, so wie es auch schon bei dem Use Case Formular möglich ist. Dies ist sinnvoll, da evtl. betriebseigene Informationen abgespeichert werden sollen. Z.B. gibt es in einem großen Unternehmen vielleicht noch einen Zweit-Durchseher. Dafür immer einen festen Punkt vorzusehen wäre für viele andere überflüssig und störend.
   - Beim Anlegen einer neuen Qualitätsanforderung kommt man in der ConstructionView in einen ‚vorgeschalteten Bereich', in dem der erste Teil des Formulars (also allgemeines und konkretes Qualitätsmerkmal) eingegeben werden muss und erst dann kann man auf einen ‚Erstellen-Button' klicken und die Qualitätsanforderung wird angelegt. Das Ändern des oberen Bereiches ist dann im ‚normalen' Formular nicht mehr möglich. Es sollte dann allerdings einen Button geben, um wieder in den ‚vorgeschalteten Bereich' zu gelangen. Dadurch wird der Benutzer dann wirklich gezwungen, sich beim Ausfüllen an die Reihenfolge im Formular zu halten, und jede Qualitätsanforderung kann so auch nach allgemeinen Qualitätsmerkmalen sortiert werden.
8. Erfahrungsbasis verwalten:
   - Das Erstellen von Warnungen und Fehlern ohne die Kenntnis von gewissen Java-Klassen ermöglichen. Dazu müssten die Fehler und Warnungen in gewisse Kategorien eingeteilt werden, die dann z.B. durch ein Push-Down-Menü ausgewählt werden können, anstatt dass ein Klassenname eingegeben werden muss. (Spart Zeit beim Erstellen und ermöglicht diese Funktion auch quelltextfremden Nutzern.)

[16] [Cri06] S. 24

5 Fazit

Die Umsetzung der Anforderungen an den Editor wurde erfüllt. Alle bisherigen Funktionen für Use Cases blieben erhalten und die neu geforderten können fehlerfrei durchgeführt werden. Dazu gehört das Erstellen, Speichern und Laden von Projekten und das Erstellen, Bearbeiten und Löschen von Qualitätsanforderungen. Es können beliebig viele Qualitätsmaßstäbe und zu jedem Qualitätsmaßstab beliebig viele Soll-Werte und Prüfungsdurchführungen hinzugefügt und wieder gelöscht werden. Die Qualitätsanforderungen können als HTML-Tabelle exportiert werden und mit denen in der Vorlesung „Software-Qualität“ benutzen Farben des Qualitätsmodells hinterlegt werden. Nur das Sortieren im Projekt-Baum nach weiteren Kriterien, zusätzlich zum Namen des konkreten Qualitätsmerkmals, ist nicht möglich. Das Anzeigen, Erstellen, Bearbeiten, Löschen, Kommentieren und Ignorieren der Kritiken ist für die Qualitätsanforderungen genauso möglich, wie bisher schon für die Use Cases. Der Editor ist außerdem abwärtskompatibel, denn auch die mit dem alten Editor erstellten Projekte können geladen werden. Dort stehen dann ebenfalls alle neuen Funktionen zur Verfügung.

Die erfolgreiche Umsetzung der qualitativen Anforderungen kann jetzt durch die Rückmeldung mehrerer Benutzer ermittelt werden, die bestenfalls auch zum vorherigen Editor noch keinen Kontakt hatten, da diese Qualitätsanforderungen als Qualitätsziel das Qualitätsmerkmal ‚Benutzbarkeit’ mit Schwerpunkt ‚Erlernbarkeit’ haben.

Es wäre wünschenswert, wenn sich der Editor in Zukunft einem ‚Praxis-Test’ unterzöge, indem er z.B. vom Fachgebiet Software Engineering in den künftigen Software-Projekten eingesetzt würde und die beteiligten Leute den gezogenen Nutzen beurteilten. Schließlich ‚lebt’ der Editor von der gewonnenen und schon vorhandenen Erfahrung der Anwender, die dieses Wissen in die wachsende Erfahrungsbasis einfügen, wodurch die Erstellung besserer Qualitätsanforderungen an eine Software ermöglicht wird.

Literaturverzeichnis

[Boe81] Boehm, Barry: *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice Hall, 1981.

[Coc01] Alistair Cockburn: *Writing effective use cases*. Amsterdam: Addison Wesley, 2001

[Cri06] Crisp, Christian: *Konzept und Werkzeug zur erfahrungsbasierten Erstellung von Use Cases.* Masterarbeit, Leibniz Universität Hannover, 2006.

[Fow99] Fowler, Martin: *Refactoring: Improving the Design of Existing Code*. Amsterdam: Addison Wesley, 1999.

[Lig02] Liggesmeyer, Peter: *Software-Qualität*. Heidelberg, Berlin: Spektrum, Akad. Verlag., 2002.

[Sch06] Schneider, Kurt. *Software-Qualität.* Vorlesungsscript, Leibniz Universität Hannover, 2006.

[Vsek07] Virtuelles Software Engineering Kompetenzzentrum, Glossareintrag: *Qualitätsmodell*, online 15.02.07, http://www.software-kompetenz.de/?11894.

[Wik07] Wikipedia – die freie Enzyklopädie, Artikel: *ISO/IEC 9126*, online 08.02.07, http://de.wikipedia.org/wiki/ISO/_IEC_9126, Artikel: *Refactoring*, online 08.02.07, http://de.wikipedia.org/wiki/Refactoring.

Anhang

- Weitere funktionale und qualitative Anforderungen für den erstellten Editor befinden sich auf der CD, sowohl als Projektdatei für den Editor, als auch als exportierte HTML-Datei.
- Außerdem befinden sich auf der CD der Quelltext des Editors, sowie diese Studienarbeit selbst.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Studienarbeit selbstständig und ohne fremde Hilfe verfasst habe und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe.

_____

Hannover, 19.02.2007 Stefan Keil