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

Quantitative und qualitative Analyse von Anforderungen in Software-Projekten

Studienarbeit

im Studiengang Mathematik mit Studienrichtung Informatik

von

Nicko C.-W. Horst

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

Hannover, Februar 2007

Erklärung

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

Hannover, 15.02.2007 Nicko C.-W. Horst

„Um was für eine Liebesgeschichte handelt es sich denn?“

„Jetzt werd’ bloß nicht sentimental. Das haben wir hinter uns. Wir haben die Sentimentalität überwunden.“

„Ich frag’ doch nur. *Ich sammele Fakten. Nackte kalte Fakten.*“

„Es handelt sich um eine Liebe, die über den Tot hinausgeht.“

„Tatsächlich? Wie romant.. äh ich meine ... erzähl mehr! *Mehr nackte, kalte Fakten.*“

Walter Moers : Rumo und die Wunder im Dunkeln

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Ein großes Problem bei der Planung von Software-Projekten ist die Abschätzung des zeitlichen Aufwandes. Hierzu wird zum Beginn des Projektes eine Anforderungsspezifikation erstellt, die das Ziel des Projektes und alle Anforderungen, die an das zu entwickelnde System gestellt werden, beinhaltet. Des Weiteren enthält das Dokument Prioritäten zur Umsetzung der Anforderungen und mögliche Einschränkungen und Ausbauvarianten. Die Schwierigkeiten bestehen dabei zu garantieren, dass alle nötigen Anforderungen erfasst sind und der Abschätzung der Zeit, die benötigt wird um alle an das System gestellten Anforderungen umzusetzen, die in der Anforderungsspezifikation festgehalten sind.

In Softwarefirmen ist es üblich für jede Anforderung aus der Anforderungsspezifikation eine Zeitabschätzung zu machen, die angibt wie lange die Implementierung dieser Anforderung ungefähr dauert. Dabei kann der Fall eintreten, dass die Anforderung nicht ausreichend spezifiziert war und die Implementierung aufwendiger ist als angenommen. Dann muss die Anforderung näher untersucht, aufgeteilt und der Zeitaufwand zur Implementierung erneut abgeschätzt werden.[1]

Die Fragen sind nun: Kann man Anforderungen so formulieren, dass mögliche Unzulänglichkeiten einer Anforderung bereits bei der Formulierung auffällig werden? Und kann man eventuell bereits über die Anzahl der Anforderungen, Unzulänglichkeiten derselben feststellen oder gar eine Aussage über den benötigten Zeitaufwand zur Implementierung machen?

1.2 Aufgabenstellung

Im Rahmen dieser Studienarbeit sollen die Anforderungsspezifikationen aller Software-Projekte des Wintersemesters 2006/2007 bezüglich ihrer enthaltenen Anforderungen quantitativ und qualitativ analysiert werden.

Ziel der Studienarbeit ist es, mit Hilfe der gewonnenen Daten qualitative Aussagen über die Anforderungen der bearbeiteten Software-Projekte zu machen.

1.3 Aufbau der Arbeit

Die Studienarbeit gliedert sich im Wesentlichen in zwei Bereiche: Die quantitative und die qualitative Analyse der Anforderungsspezifikationen der Software-Projekte 2006/2007.

Bei der quantitativen Analyse sollen Anforderungen und Anwendungsfälle für einen späteren Vergleich gezählt werden. Dafür muss zuerst geklärt werden, welche Möglichkeit es gibt Anforderungen und Anwendungsfälle zu zählen.

[1] Vgl. Anhang A

Bei der qualitativen Analyse sollen dann die in der quantitativen Analyse gewonnen Daten verwendet werden, um Vermutungen über den Zusammenhang von Anforderungen und Anwendungsfällen zu überprüfen.

2 Quantitative Erhebung

2.1 Suchen und Finden von Anforderungen

Einen zentralen Punkt der Anforderungsspezifikationen bilden die Anwendungsfälle, da diese alle Schritte beinhalten, die der Benutzer während der Arbeit mit dem System immer wieder ausführt. Eine treffende Definition von Oestereich lautet: „Ein Anwendungsfall beschreibt eine Menge von Aktivitäten, die aus der Sicht der Endanwender zu einem wahrnehmbaren Ergebnis führen.“[2] Nach Jacobson, dem Erfinder der Use-Cases, ist die Arbeit mit Anwendungsfällen ein Instrumentarium für die Definition von Anforderungen.[3] Die Idee liegt darin, dass bei der Beschreibung der Anwendungsfälle, alle Anforderungen, die der Benutzer an das System stellt, bereits mit formuliert werden.

Daher wurden im Rahmen dieser Studienarbeit zunächst alle Anwendungsfälle der Anforderungsspezifikationen der einzelnen Softwareprojekte betrachtet, um alle dort formulierten Anforderungen zu suchen. Im weiteren Verlauf wurden dann die restlichen Abschnitte der Anforderungsspezifikationen auf weitere Formulierungen von Anforderungen durchsucht, falls doch nicht alle Anforderungen ausschließlich in den Anwendungsfällen beschrieben wurden.

Ein größeres Problem bestand im Auffinden der Anforderungen, da bei der Formulierung der Anforderungsspezifikationen durch die Verwendung von Sprache Informationen transportiert, weggelassen, verallgemeinert oder verzerrt wurden.

Einige Beispiele aus den bearbeiteten Software-Projekten:

- *„Eine Kalibrierung soll nur einmal pro Schicht nötig sein.“*
  Was soll kalibriert werden?
  Die Frage ist auch unter Verwendung der vollständigen Textpassage nicht eindeutig zu beantworten.
- *„Ein ausführliches Handbuch soll vorhanden sein.“*
  Was bedeutet in diesem Fall „ausführlich“?
- „*Server verwendet neue Benutzerdaten.“*
  Was bedeutet „verwenden“?

2.2 Formulieren der Anforderungen

Die formulierten Anforderungen sind durch die Sprache nicht nur ungenau, verallgemeinert oder verzerrt, sondern weisen auch noch andere Mängel auf. In den meisten Fällen sind sie unvollständig, nicht direkt umsetzbar oder testbar und die juristische Verbindlichkeit ist nicht erkennbar. Nach Helmut Partsch gibt es keine Formalismen oder Konstrukte zur Formulierung nicht-funktionaler

[2] Vgl. [Oes98]
[3] Vgl. [JCJO92]

Anforderungen.[4] Die Autoren Rupp et al. beschreiben allerdings eine Möglichkeit mit Hilfe von Anforderungsschablonen funktionale Anforderungen zu beschreiben und dabei die Unzulänglichkeiten der Sprache zu beheben.[5] Anforderungsschablonen erzwingen für die Anforderungen eine ähnliche Struktur, und versuchen so typische Formulierungsfehler zu verhindern.

In Abbildung 1 wird die Kernstruktur einer Anforderungsschablone dargestellt, die zunächst aus den zwei Blöcken „Rechtliche Verbindlichkeit“ und „Systemaktivität“ besteht.

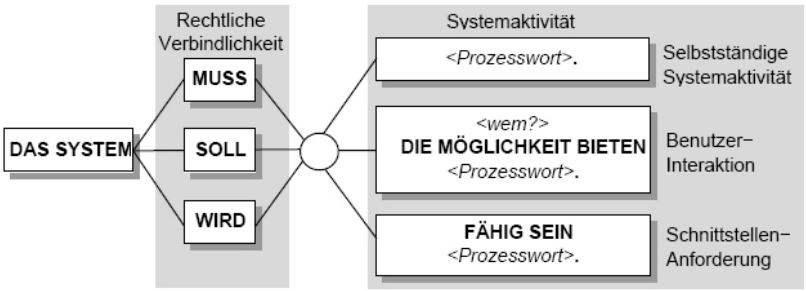

Abbildung 1: *Grundform der Anforderungsschablone*[6]

Dabei wird die Systemaktivität in drei Teilbereiche unterteilt, womit die meisten funktionalen Anforderungen an das System abzubilden sind.[7] Das Erstellen der Anforderung mit Hilfe der Schablone erfolgt in fünf Schritten, wobei die Grundform der Schablone um Vor- und Nachbedingungen erweitert wird.

1. Identifizieren des Prozesses
   Die geforderte Funktionalität der Anforderung wird durch das Prozesswort definiert und ist ein Verb, das den gesamten grammatikalischen Bau des Satzes beeinflusst. Der Prozess charakterisiert den funktionalen Aspekt einer Anforderung.
2. Festlegen der rechtlichen Verbindlichkeit
   - muss - Die Anforderung ist rechtlich bindend.
   - soll - Die Anforderung ist dringend empfohlen.
   - wird - Die Anforderung ist zukünftig bindend.
3. Charakterisieren der Aktivität im System
   - Das System führt den Prozess selbstständig durch. (**Abbildung 1**: Selbstständige Systemaktivität)
   - Das System stellt dem Benutzer die Prozessfunktionalität zur Verfügung. (**Abbildung 1**: Benutzerinteraktion)

[4] Vgl. [Par98]
[5] Vgl. [RD00], [RD01], [R+02]
[6] In Anlehnung an [R+02] S.236.
[7] Vgl. [R+02] S.232 für eine Vollständige Herleitung.

- Das System führt den Prozess in Abhängigkeit eines Dritten aus, ist passiv und wartet auf den externen Auslöser. (**Abbildung 1**: Schnittstellenanforderung)

4. Feinschliff des Prozesses
   Bisher wurde lediglich die Grundform einer Anforderung konstruiert. Anforderungen werden neben dem Prozesswort, der rechtlichen Verbindlichkeit und der Systemaktivität häufig durch Objekte und Ergänzungen des Objektes beschrieben. Es fehlt bisher die nähere Bestimmung des Prozesswortes.
5. Formulieren der logischen und zeitlichen Bedingungen
   Bei technischen Systemen müssen alle logischen Vor- und Nachbedingungen, die häufig auch zeitlich variieren, berücksichtigt werden. Diese als Randbedingungen bezeichneten Aspekte werden an den Anfang einer Anforderung gesetzt.

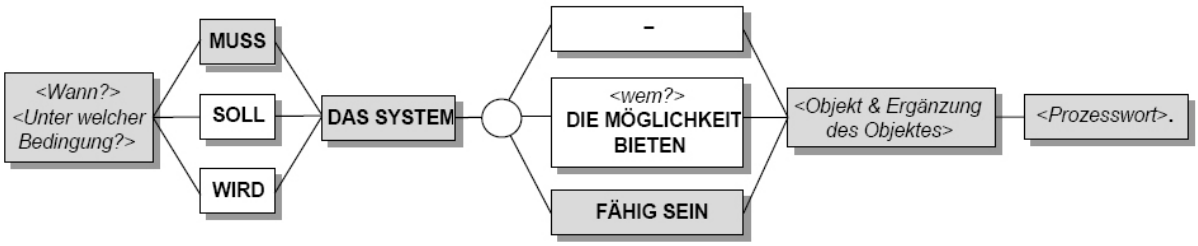

Abbildung 2: *Vollständige Anforderungsschablone*[8]

Abbildung 2 zeigt die vollständig konstruierte Anforderungsschablone, die um die Schritte vier („Objekt & Ergänzung des Objektes“) und fünf („Unter welcher Bedingung?“) ergänzt wurde. Die syntaktische Abwandlung der Reihenfolge ist hierbei durch die Eigenschaften der deutschen Sprache begründet. Ebenso mögliche Abweichungen beim Prozesswort durch ein pronominales Objekt.

Diese Schablone, entwickelt um Anforderungen von vornherein korrekt zu formulieren, wurde in dieser Studienarbeit dazu verwendet um alle Anforderungen in den Anforderungsspezifikationen in eine einheitliche Form zu bringen. Diese bietet zum einen den Vorteil, dass bei dem Versuch einer Neuformulierung sich leichter entscheiden lässt bei welchen Textpassagen es sich überhaupt um eine Anforderung handelt, und zum anderen, dass die Anforderungen in dieser Form leichter zu vergleichen sind und somit mehrfach auftretende Anforderungen zu einer zusammengefasst werden können. So bearbeitet können die Anforderungen einer Anforderungsspezifikation einfach gezählt werde, indem man die Anzahl der benötigten Anforderungsschablonen zählt.

[8] Vgl. [R+02] S.238

2.3 Auftretende Probleme

Bei der Vereinheitlichung bereits bestehender Anforderungen mit Hilfe der Anforderungsschablone nach Rupp et al. kommt es mehrfach zu Problemen. Die Probleme sind größtenteils auf unzureichendes Fachwissen im Bereich des betreffenden Software-Projekts oder mangelhafte Formulierung der ursprünglichen Anforderung zurückzuführen. Die auftretenden Probleme lassen sich in zwei Kategorien einteilen: In Probleme bei der Neuformulierung der Anforderungen und in Probleme beim Zusammenführen zweier inhaltlich gleicher Anforderungen.

2.3.1 Formulierungsprobleme

Während der Neuformulierung der Anforderungen und Vereinheitlichung mit Hilfe der Anforderungsschablone kommt es an verschiedenen Stellen zu Problemen. Das größte Problem besteht in der Abschätzung der rechtlichen Verbindlichkeit der Anforderung. Zwar werden am Anfang jedes Dokumentes die Prioritäten des Kunden in einer Liste mit Absteigender Priorität aufgeführt, jedoch ist dort keine feste Einteilung zu finden. Somit entstammen die Angaben zur *rechtlichen Verbindlichkeit* in den neu formulierten Anforderungen einer groben Einschätzung, abhängig davon, ob die Anforderung in der Prioritätenliste relativ weit oben oder unten steht. Hilfreich bei der Einschätzung sind vereinzelte Kommentare innerhalb des Dokumentes, in denen auf die ‚mögliche Implementierung bei ausreichender Zeit' hingewiesen wird. Es wurde entschieden so gekennzeichnete Anforderungen, zusammen mit Anforderungen, die lediglich in Erweiterungen von Anwendungsfällen auftreten und den Anforderungen am Ende der Prioritätenliste, als *dringend empfohlen* einzustufen. Die restlichen Anforderungen der Prioritätenliste wurden als *rechtlich bindend* eingestuft.

Ein weiteres Problem besteht in dem teilweise unzureichenden Fachwissen auf dem Gebiet des zu bearbeitenden Software-Projektes. Wenn es sich dabei lediglich um die Unkenntnis einzelner Fachwörter handelt, kann die Formulierung einfach übernommen werden, da der Inhalt der Anforderung für eine quantitative Erhebung nicht von belang ist. Diese Entscheidung wirft allerdings Schwierigkeiten an anderer Stelle auf. Besonders wenn die unbekannten Fachwörter den funktionalen Aspekt des Systems charakterisieren. (*siehe 2.3.2*) Teilweise konnte bei der Neuformulierung jedoch nicht entschieden werden, ob es sich bei der vorliegenden Formulierung um eine an das System gestellte Anforderung handelt oder nicht. In diesen Fällen wurde auf eine Formulierung verzichtet.

Auch gibt es Fälle in denen die formulierte Anforderung logisch nicht umsetzbar ist. In diesen Fällen wurde der Fehler in die neu formulierte Anforderung mit übernommen, da der Inhalt der Anforderungen für diese Studienarbeit nicht relevant ist.

Einige Beispiele aus den bearbeiteten Software-Projekten:

- *„Der Editor soll als Erweiterung (Plugin) des ProFLOW-Frameworks als Eclipse-Plugin erstellt werden.“*[9]
  An dieser Stelle scheiterte die Formulierung an Unkenntnis über die Verwendung von Plugins bei Eclipse. Es kann nicht entschieden werden, ob es sich um eine funktionale Anforderung handelt.
- *„Die FTS dürfen im Betrieb einen Sicherheitsabstand zueinander nicht unterschreiten.“*[10]
  Auch typisch: Das formulieren von Situationen, die *nicht* eintreten sollen. In diesem Fall muss eine größere Transferleistung erbracht werden, damit die Anforderung dem Schema der Anforderungsschablone entspricht. Eventuell müssen sogar mehrere Schablonen verwendet werden. Z.B.:
  In Bewegung | muss | das FTS | den Abstand zu anderen FTS | berechnen. Unterschreitet der Abstand einen Sicherheitswert | muss | das FTS | ...[11]
- *„Es soll eine menschlich lesbare XML-Matching-Statistik erstellt werden.“*[12]
  Wie groß ist die Bedeutung von „menschlich lesbar“ in diesem Fall? Kann dieser Zusatz einfach weggelassen werden?

2.3.2 Probleme beim Zusammenfassen

Auch beim Zusammenfassen mehrer Anforderungen gleichen Inhalts treten Probleme auf. Zwar wird diese Arbeit durch die ähnliche Struktur der mit Hilfe der Schablone formulierten Anforderungen vereinfacht, jedoch treten Situationen auf in denen eine Zusammenfassung nicht möglich ist.

So kommt es vor, dass von zwei Anforderungen, die sich nur in einem Punkt unterscheiden, nicht entschieden werden kann, ob sie gleichen Inhalts sind oder nicht. Dies liegt häufig daran, dass der Unterschied in einem Fachwort liegt, das nicht bekannt ist und somit von der Bedeutung her nicht mit anderen, bekannten Wörtern verglichen werden kann. Auch kann es auftreten, dass zwei Anforderungen widersprüchlich sind. In diesem Fall können die Anforderungen in Hinsicht auf eine quantitative Erhebung zwar zusammengefasst werden, jedoch kann sich nur willkürlich für eine der beiden Anforderungen entschieden werden.

Bei *eindeutig bedingten Abläufen*, d.h. unter Bedingungen auftretenden Anforderungen, die auf eine bestimmte andere Anforderung beschränkt sind, bestünde die Möglichkeit diese Abläufe von Anforderungen zu einer einzigen Anforderung zusammenzufassen. Wahrscheinlich ist jedoch, dass diese Bedingun-

[9] Aus SOA-Me Editor : Anforderungsspezifikation, Version 1.3
[10] Aus NEH-A : Anforderungsspezifikation, Revision 81
[11] Die ‚|’ trennen an dieser Stelle die einzelnen Abschnitte der Anforderungsschablone.
[12] Aus PPM : Anforderungsspezifikation, Version 1.1

gen nur deshalb eindeutig sind, weil nicht alle Anforderungen formuliert wurden. Daher wurden die Anforderungen in diesen Fällen nicht zusammengefasst.

Ein Problem anderer Art sind *Implizierte Anforderungen.* Dabei handelt es sich um Anforderungen, die beim Erstellen der Anforderungsspezifikationen nicht formuliert werden, jedoch aus dem Kontext heraus eindeutig erhoben werden. In diesen Fällen wäre es möglich die Anforderungen im Nachhinein zu formulieren, um diese bei der quantitativen Erhebung mitzählen zu können, jedoch wurde darauf verzichtet, da bei dieser Vorgehensweise keine vollständige Deckung zu garantieren ist.

Einige Beispiele aus den bearbeiteten Software-Projekten:

- *Wenn Daten eingegeben werden | muss | das System | die eingegebene Daten in Anweisungen | umsetzen.*
  Wenn Anweisungen umgesetzt werden | muss | das System | die Anweisungen in ein RCX kompatibles Format | konvertieren.
  Wenn Anweisungen konvertiert werden | muss | das System | fähig sein | die Anweisungen über den Infrarot-Port | zu versenden.[13]
  Die drei Anforderungen bilden zusammen einen eindeutig bedingten Ablauf und könnten daher zusammengefasst werde. Allerdings ist nicht klar, ob es nicht noch andere Auslöser der zweiten oder dritten Anforderung gibt.
- „[Die Lichtsensoren] *können keine Werte liefern.“*
- „[…], *dass sich ihre* [gelieferten] *Werte mit sinkender Spannung der Bord-Batterien ändern.“*[14]
  Diese zwei widersprüchlichen Aussagen haben keine Auswirkung auf die quantitative Erhebung, da sie zu einer einzigen Anforderung zusammenfallen, deren Inhalt für diese Studienarbeit nicht von Belang ist.

2.4 Besonderheiten

Neben Problemen beim Formulieren und Zusammenfassen der neuen Anforderungen traten in den bearbeiteten Anforderungsspezifikationen auch unerwartete Beschreibungsformen der von dem System geforderten Eigenschaften auf. Diese Formulierungen haben für die quantitative Erhebung keine Bedeutung, da sie nicht mit Hilfe der Anforderungsschablone in Anforderungen umzusetzen sind, jedoch weisen sie sehr gut auf, welche Fehler oder unnötige Arbeit sich mit Hilfe der Anforderungsschablone vermeiden lassen.

Einige Besonderheiten aus den bearbeiteten Software-Projekten:

- *„Die Güter müssen in den Ausgangslagern in eine definierte Reihenfolge gebracht werden.“*[15]

[13] Die ‚|’ trennen an dieser Stelle die einzelnen Abschnitte der Anforderungsschablone.
[14] Aus NEH-A : Anforderungsspezifikation, Revision 81

Hier liegt ein logischer Fehler vor, da das Sortieren der Güter *in* den Ausgangslagern nicht möglich ist. Sie können bestenfalls in einer sortierten Reihenfolge eingelagert werden.

- Es fiel auf, dass die zentrale Anforderung eines Anwendungsfalls nicht immer in dem Anwendungsfall auftritt. Zum Beispiel tritt im Use Case „Prozess aussuchen und starten“ des Projektes ‚*SOA-Me Client’* das Starten eines Prozesses als Anforderung nicht auf.
- In einigen Anwendungsfällen treten Anforderungen auf, die kein zentraler Punkt des Anwendungsfalls sind, jedoch an anderer Stelle nicht auftreten. So findet sich das Starten eines Prozesses aus dem vorhergehenden Beispiel im Anwendungsfall „Anmeldung“ wieder.

[15] Aus NEH-A : Anforderungsspezifikation, Revision 81

3 Qualitative Erhebung

3.1 Numerische Vergleiche

Die Anforderungsspezifikationen unterteilen sich aus Sicht dieser Studienarbeit in zwei wesentliche Bereiche: Die Anwendungsfälle und der Rest des Dokuments. Es wird erwartet, dass bei einer sorgsamen Ausarbeitung der Anwendungsfälle, diese alle Anforderungen, die an das System gestellt werden, beinhalten. (*Vgl. 2.1*) Alle Anforderungen, die im weiteren Dokument vorkommen, sollten beim Zusammenfassen mit Anforderungen aus den Anwendungsfällen zusammengefallen sein. Wurden weniger Anwendungsfälle ausgearbeitet, so ist zu vermuten, dass die Anzahl der Anforderungen in den Anwendungsfällen proportional mit der Anzahl der Anwendungsfälle abfällt, da diese unabhängig voneinander sind. Des Weiteren ist anzunehmen, dass die Anzahl der Anforderungen, die nur außerhalb der Anwendungsfälle auftreten, zunimmt und somit dem Informationsverlust entgegenwirkt.

3.1.1 Anforderungen in Anwendungsfällen

Tabelle 1 zeigt die Anzahl von Anwendungsfällen und darin enthaltene Anforderungen in den Anforderungsspezifikationen aller teilnehmenden Gruppen des Software-Projektes. Diagramm 1 zeigt noch einmal die Anzahl der Anforderungen in Abhängigkeit von der Anzahl der Anwendungsfälle. Tabelle und Diagramm können die geäußerte Vermutung eines proportionalen Anstiegs der Anforderungen zu den Anwendungsfällen nicht bestätigen.

Gruppe	NEH-A	NEH-B	NEH-C	PPM	SOA-Me Cl.	SOA-Me Ed.	SOA-Me Sv.	WüSin
Anwendungsfälle	6	2	9	8	6	6	8	17
Anforderungen	13	10	11	13	24	23	24	35

Tabelle 1: *Anzahl von Anforderungen in Anwendungsfällen*[16]

[16] Vgl. Anhang D.

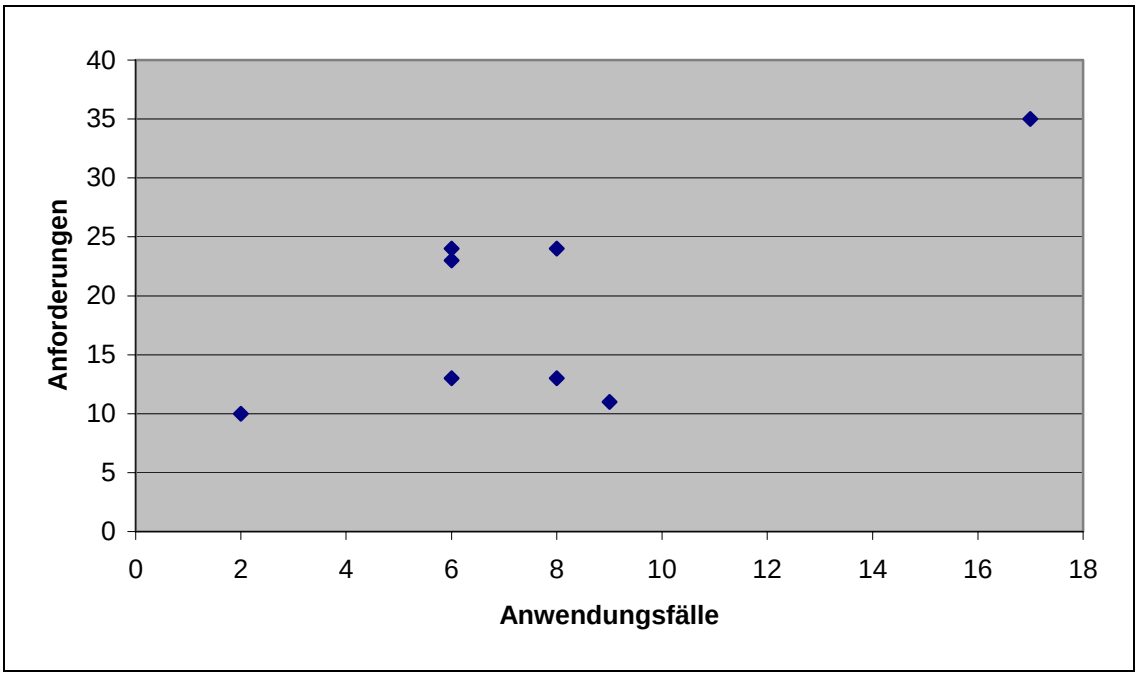

Diagramm 1: *Anzahl von Anforderungen in Anwendungsfällen*[17]

3.1.2 Anforderungen außerhalb der Anwendungsfälle

Diagramm 2 zeigt anteilsmäßig die Anforderungen, die nur innerhalb oder außerhalb der Anwendungsfälle der Anforderungsspezifikationen vorkommen. Außerdem auch die Anforderungen, die in beiden Bereichen zu finden sind.

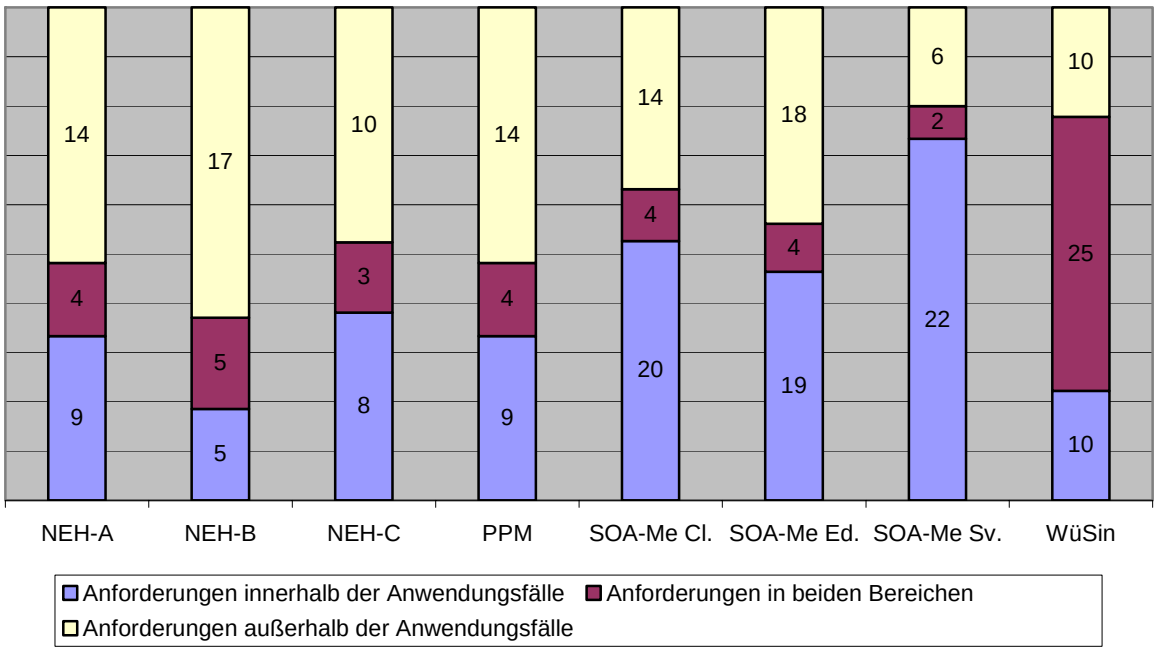

Diagramm 2: *Anforderungen innerhalb und außerhalb der Anwendungsfälle*[18]

[17] Vgl. Anhang D.
[18] Vgl. Anhang D.

Geht man von der Annahme Jacobsons aus, dass die Anwendungsfälle alle Anforderungen enthalten, die der Benutzer an das System stellt[19], so müssten die Gruppen ‚WüSin' und ‚SOA-Me' Server die beste Abdeckung von Anwendungsfällen erreicht haben, die Gruppe ‚NEH-B' die schlechteste. Aus Tabelle 1 lässt sich entnehmen, dass die Gruppe ‚WüSin' die meisten Anwendungsfälle konstruiert hat, die Gruppe ‚NEH-B' die wenigsten. Da es sich bei den behandelten Software-Projekten um lediglich acht Projekte handelt, darunter drei gleiche, lässt sich die Vermutung, dass die Anzahl der Anforderungen außerhalb der Anwendungsfälle mit abnehmender Anzahl von Anwendungen zunimmt, nicht bestätigen, jedoch widersprechen die gemachten Beobachtungen der Vermutung und auch der Annahme Jacobsons nicht.

3.2 Zusammenarbeit mehrerer Gruppen

An drei Software-Projekte mit gleicher Aufgabenstellung wurde die zusätzliche Bedingung gestellt, dass einzelne Hardware-Komponenten unter den Software-Projekten ausgetauscht werden können. Um dies zu gewährleisten wurde von den drei Gruppen gemeinsam ein Kommunikationsprotokoll erstellt, das Richtlinien für die Kommunikation der einzelnen Hardware-Komponenten beinhaltet.

Zusätzlich zu den Anforderungsspezifikationen wurden auch die Anforderungen dieses Dokuments mit der Anforderungsschablone neu formuliert, um sie in die Untersuchung dieser Studienarbeit mit einzubeziehen. Es wird vermutet, dass jede Anforderung, die im Kommunikationsprotokoll enthalten ist, auch in mindestens einer der drei Anforderungsspezifikationen auftritt.

Ein Vergleich der Anforderungen zeigt jedoch, dass nicht eine der Anforderungen aus dem Kommunikationsprotokoll bereits in einer der Anforderungsspezifikationen auftrat. Dies liegt daran, dass in dem Kommunikationsprotokoll nur hardwarebedingte Anforderungen formuliert wurden, die spezifischer sind als Anforderungen aus den Anforderungsspezifikationen.[20]

[19] Vgl. [JCJO92]
[20] Vgl. Anhang B.

4 Zusammenfassung

Ziel dieser Studienarbeit war es qualitative Aussagen über die Anforderungen der bearbeiteten Software-Projekte zu machen. Welche Aussagen können getroffen werden?
Die quantitative Erhebung zeigt, dass es mit Hilfe der Anforderungsschablone nach Rupp et al. möglich ist qualitativ hochwertige Anforderungen zu formulieren, die die Umsetzung des Software-Projektes erleichtern, da es mit ihnen einfacher ist, die Liste der Anforderungen auf Fehler und Unvollständigkeit zu überprüfen und Anforderungen weiter aufzuteilen, um deren Zeitaufwand bei der Umsetzung besser einschätzen zu können.
Die Gespräche mit den Software-Firmen, die im Rahmen dieser Studienarbeit stattgefunden haben, zeigen wie wichtig eine vollständige Anforderungsliste ist, um einen reibungslosen Ablauf während der Implementierungsphase zu gewährleisten. Sollte also in späteren Software-Projekten gefordert werden die Anforderungen nach einem Anforderungsschablonen ähnlichem Prinzip zu formulieren, so dürfte das die Übersichtlichkeit der Projekte, sowohl für den Betreuer, als auch für die Studenten, steigern und somit die Umsetzung erleichtern.[21]
Die quantitative Erhebung zeigt, dass in der gewählten Form der Analyse viel Potential steckt, um die Bedeutung der Anwendungsfälle zur Formulierung einer vollständigen Anforderungsliste zu ergründen. Jedoch war die Anzahl der Betrachteten Software-Projekte viel zu gering und die einzelnen Software-Projekte zu verschieden, um klare Schlüsse ziehen zu können. Des Weiteren wäre eine Betrachtung weiterer Dokumente aus dem fortgeschrittenen Verlauf der Projekte sinnvoll, da sich mit ihnen weitere Einblicke in die Vollständigkeit der Anwendungsfälle gewinnen ließen.

Am Ende war die quantitative und qualitative Analyse der Software-Projekte ein Schritt in die richtige Richtung. Jedoch war der Schritt zu klein. Man braucht mehr —

„Mehr nackte, kalte Fakten.“[22]

[21] Vgl. Anhang A.
[22] Walter Moers : Rumo und die Wunder im Dunkeln. 5. Auflage : Piper Verlag, 2004

Literatur

[JCJO92] Jacobson, Ivar ; Christerson, Mahnus ; Jonsson, Patrik ; Overgaard, Gunnar: *Object-Oriented Software Engineering*. Reading, MA : Addison-Wesley, 1992

[Oes98] Oestereich, Bernd: *Objektorientierte Softwareentwicklung – Analyse und Design mit der UML*. 4., aktualisierte Auflage. München, Wien : Oldenburg Verlag, 1998

[Par98] Partsch, Helmut: *Requirements-Engineering systematisch*. 1. Auflage. Berlin : Springer Verlag, 1998

[R+02] Rupp, Chris et al.: *Requirements-Engineering und -Management*. 2., überarbeitete Auflage. München, Wien : Carl Hanser Verlag, 2002

[RD00] Rupp, Chris ; Dallner, Jürgen: Requirements-Engineering – der Einsatz einer natürlichsprachlichen Methode bei der Ermittlung und Qualitätsprüfung von Anforderungen. In: *OBJEKTspektrum* Nr. 2 (Januar 2000)

[RD01] Rupp, Chris ; Dallner, Jürgen: Mustergültige Anforderungen. In: *OBJEKTspektrum* Nr. 3 (März 2001)

Anhang

A. Vergleich mit realen Software-Projekten

Um einen Vergleich zwischen den Software-Projekten der Leibniz Universität Hannover und denen realer Software-Firmen ziehen zu können wurden in Rahmen dieser Studienarbeit Gespräche mit Angestellten von zwei Software-Firmen geführt. Mit Angestellten der Firmen *Nik Software Inc.* und *Reakktor Media*. Im Rahmen dieser Gespräche wurde unter anderem der dort übliche Ablauf beim Aufstellen und Prüfen der Anforderungen von Software-Projekten besprochen, sowie ein Einblick in die Organisationsstruktur gewährt.

B. Neu formulierte Anforderungen

Alle neu formulierten Anforderungen liegen als Anlage auf der CD im Text-Format im Ordner ‚Anforderungen' vor. Die einzelnen Abschnitte der Anforderungsschablone sind dabei mit ‚|' voneinander getrennt.

C. Zusammengefasste Anforderungen

Die zusammengefassten Anforderungen liegen als Anlage auf der CD im Text-Format im Ordner ‚Anforderungen - Merged' vor und sind in Anforderungen innerhalb und außerhalb der Anwendungsfälle unterteilt. Die einzelnen Abschnitte der Anforderungsschablone sind dabei mit ‚|' voneinander getrennt.

D. Tabelle über Anzahl der Anforderungen

Eine ausführliche Tabelle über die Anzahl der Anforderungen in den Anforderungsspezifikationen liegt als Anlage auf der CD im Exel-Format vor.