Asynchrone Anwenderbeteiligung in Software-Projekten

Zur Erlangung des akademischen Grades eines
Doktors der Wirtschaftswissenschaften

(Dr. rer. pol.)

von der Fakultät für
Wirtschaftswissenschaften
des Karlsruher Institut für Technologie (KIT)

genehmigte

DISSERTATION

von

Dipl.-Inform. Asarnusch Rashid

Tag der mündlichen Prüfung:	24. Januar 2011
Referent:	Prof. Dr. Andreas Oberweis
Korreferent:	Prof. Dr. Peter Lockemann

Karlsruhe, 2011

Zusammenfassung

In dieser Arbeit wird die neue Methode OpenProposal zur asynchronen Anwenderbeteiligung in Software-Projekten entwickelt und im Rahmen von vergleichenden Fallstudien evaluiert. Kern der Methode stellt ein Annotationssystem zur asynchronen Erstellung und Bearbeitung von Anwenderbeiträgen dar, das zu einer höheren Effizienz und Effektivität der Anwenderbeteiligung in Software-Projekten verhelfen soll. Mit der neuen Methode werden die Möglichkeiten zur asynchronen Kommunikation zwischen Anwender, Entwickler, Moderator und Entscheider aufgezeigt und Vorteile gegenüber einer synchronen Kommunikation nachgewiesen. Dem neuen Ansatz liegt die Arbeitshypothese zugrunde, dass Anwender eine effizientere und einfachere Methode zur Erstellung von Anwenderbeiträgen benötigen, bei der sich qualitativ hochwertige Anwenderbeiträge ergeben. Hierfür wird der Einsatz von Annotationssystemen als produktivitätsförderndes Kommunikationssystem motiviert, mit denen Anwender die grafische Oberfläche von Softwaresystemen annotieren und als Mitteilung an das Entwicklerteam senden können.

Der erste Teil der vorliegenden Arbeit erarbeitet ein Modell zur Bewertung von Methoden zur Anwenderbeteiligung. Dieses Modell erfasst die Auswirkungen von Methoden zur Anwenderbeteiligung auf die Effizienz und Effektivität in einem Software-Projekt, beschreibt die beteiligten Rollen in einem Softwareprojekt und identifiziert die Qualität der Anwenderbeiträge, die Effizienz der Entwicklung und die Belastung der Anwender als zentrale Messgrößen.

Im zweiten Teil werden die Potentiale und Grenzen von synchronen Methoden zur Anwenderbeteiligung anhand der Bewertungskriterien aufgezeigt. Die Untersuchungen zeigen auf, dass die Potentiale von synchronen Methoden in der hohen Qualität der Anwenderbeiträge und der niedrigen Belastung der Anwender liegen, während die Effizienz der Entwicklung sich allerdings als niedrig herausstellt.

Der dritte Teil identifiziert in der Effizienz der Entwicklung das Potential von Methoden zur asynchronen Anwenderbeteiligung. Allerdings führen diese auch zu einer verringerten Qualität der Anwenderbeiträge und einer erhöhten Belastung der Anwender.

Im vierten Teil der Arbeit wird die neue Methode OpenProposal zur asynchronen Anwenderbeteiligung konzipiert, es werden Anforderungen für das Annotationssystem definiert und eine Beispielimplementierung erstellt. Im Rahmen von Usability Tests und vergleichenden Fallstudien in zwei Software-Projekten werden das Annotationssystem und die Methode evaluiert. Es wird aufgezeigt, dass OpenProposal zu einer hohen Effizienz der Entwicklung, zu einer hohen Qualität der Anwenderbeiträge und zu einer niedrigen Belastung der Anwender führt. Allerdings zeigt sich auch, dass die Methode nur für spezielle Einsatzbereiche angewendet werden kann.

Vorwort

Die Arbeit stellt die Ergebnisse der Untersuchungen zur asynchronen Anwenderbeteiligung in Software-Projekten dar. Sie führt den aktuellen Stand der Forschung zu Methoden und Werkzeugen der Anwenderbeteiligung zum Zeitpunkt der Untersuchungen auf, identifiziert Hindernisse der Anwenderbeteiligung und zeigt Verbesserungsmöglichkeiten auf. Im Sinne einer anwendungsnahen Forschung wurde mithilfe wissenschaftlicher Methoden und einer theoriegeleiteten Prototypen-Entwicklung die Methode OpenProposal entwickelt, die die Anwenderbeteiligung in Software-Projekten verbessern und damit eine Lücke in der Werkzeug- und Methodenlandschaft des Software-Engineering schließen soll. In erster Linie ist diese Arbeit somit an Wissenschaftler gerichtet, die sich mit Anforderungsanalyse bzw. Requirements Engineering, Usability Engineering oder Software-Test beschäftigen und die Bedeutung der Anwenderbeteiligung in Software-Projekten untersuchen bzw. Methoden und Werkzeuge zur Anwenderbeteiligung entwickeln.

Auf der Internetseite www.openproposal.de stehen der Prototyp OpenProposal zur kostenfreien Verwendung und auch die zum Thema veröffentlichten Publikationen zur Verfügung.

In dieser stark sozio-technisch geprägten Arbeit werden eine Vielzahl an Rollen und Berufsgruppen beschrieben, die sowohl von Frauen als auch von Männern besetzt werden können. Aufgrund der Lesbarkeit und Lesegewohnheit habe ich mich dafür entschieden, Bezeichnungen in der dritten Person Singular in ihrer männlichen Form zu wählen[1].

Karlsruhe, 06.04.2011

Asarnusch Rashid

[1] Vgl. Balzert [2000a, S. 13]

Danksagung

Ganz herzlich möchte ich mich bei meinem Doktorvater Prof. Dr. Andreas Oberweis bedanken. Seine ständige Bereitschaft und kontinuierliche Unterstützung waren eine große Bereicherung für meine Arbeit. Prof. Dr. Dr. h.c. Wolffried Stucky bin ich für seine tatkräftige Unterstützung bei meinen Tätigkeiten am FZI sehr dankbar. Ein besonderer Dank gilt Prof. Dr. Peter Lockemann für Übernahme des Koreferats und seinen unermüdlichen Einsatz in seiner Rolle als Projektleiter des Forschungsprojektes CollaBaWü. Prof. Dr. Stefan Nickel und Prof. Dr. Jan Kowalski danke ich für die Begleitung meiner mündlichen Prüfung.

Prof. Dr. Christof Weinhardt danke ich für die organisatorische Betreuung in seiner Abteilung und für die Schaffung der Freiräume für meine Forschung. Ebenso möchte ich mich bei Dr. Carsten Holtmann, Dr. Clemens van Dinther und Dr. Valentin Zacharias danken, die mich als direkte Vorgesetzte tatkräftig in meiner beruflichen Laufbahn unterstützen und eine sehr angenehme und konstruktive Arbeitsatmosphäre geschaffen haben.

Ohne die tatkräftige Unterstützung durch die zahlreichen beteiligten Studenten wäre diese Arbeit undenkbar gewesen. Eine besondere Rolle dabei nimmt David Meder ein, der von der ersten Idee, den ersten Zeilen des Quellcode und den Evaluationen bis zur Finalisierung die Arbeiten über mehrere Jahre begleitete. Mein weiterer Dank gilt Jan Baumann, Engin Kayali, Sebastian Mast, Herbert Schäfler, Monika Tavas und Ronald Walenda.

Bei Florian Gand möchte ich mich für sein Talent für grafisches Design und Kreativität bedanken. Das Logo und die grafische Oberfläche von OpenProposal stammen aus seiner Hand.

Außerdem danke ich allen Kolleginnen und Kollegen am FZI und den an den Fallstudien beteiligten Personen für die gute Zusammenarbeit und ihre Hilfsbereitschaft. Mein besonderer Dank gilt Dr. Astrid Behm, Dr. Helmuth Elsner, Hans-Jörg Happel, Athanasios Mazarakis und Tim Romberg, die mit ihren zahlreichen Hinweise hilfreich zur Seite standen. Außerdem möchte ich allen Beteiligten danken, die OpenProposal entweder in den Fallstudien oder in den Anwendertests getestet haben und zahlreiche konstruktive Hinweise einbrachten: Hans-Jörg Happel, Uwe Kippnich, Laura Plonka, Dr. Sandra Riedewald, Thomas Schuster, Peter Szulman, Dr. Volker Ziegler, Tina Müller-Arnold, Dr. Michael Rathgeb, Simone Braun, Stephan Stathel, Mercé Müller-Gorchs, Dr. Stephan Grimm, Heiko Paoli, Kerstin Schmidt, Dr. Clemens van Dinther, Dr. Mark Hefke, Dominik Gallus, Wolfgang Müller, Dr. Andreas Walter, Dr. Christian Bartsch, Daniel Rief, Sebastian Ortlieb, Dr. Valentin Zacharias, Heiko Paoli, Florian Schlüfter, Dr. Arun Anandasivam, Dr. Jochen Stößer und Dr. Carsten Block.

Die Ergebnisse dieser Arbeit entstanden während meiner Tätigkeit am FZI Forschungszentrum Informatik im Rahmen der Forschungsprojekte CollaBaWü und CollaBaWü Plus. Dafür möchte ich mich bei den Mitarbeiterinnen und Mitarbeitern der Landesstiftung Baden-Württemberg und der MFG Medien- und Filmgesellschaft Baden-Württemberg für die Finanzierung und Betreuung der beiden Projekte danken. Außerdem gilt ein großer Dank den Projektpartnern der Universität Mannheim (Lehrstühle von Prof. Dr. Armin Heinzl und Prof. Dr. Martin Schader) und dem Karlsruher Institute of Technolo-

gy (ehem. Universität Karlsruhe, Lehrstuhl von Prof. Dr. Thomas Dreier) für die angenehme Zusammenarbeit.

Meinen Eltern und Brüdern danke ich, dass sie immer für mich da sind und mich in allen Lebensphasen mit allen zur Verfügung stehenden Möglichkeiten unterstützen. Bei meinem Bruder Maz Rashid möchte ich mich darüber hinaus für seine wertvollen Beiträge aus Sicht eines Beraters, Projektmanagers und Softwareingenieurs mit langjährigen Erfahrungen in Software-Projekten bedanken.

Meinen Freunden, insbesondere Britta, Christina, David, Dominik, Hans-Peter, Holger, Jakob, Janette, Katrin, Marc, Nils, Sebastian und Werner, danke ich für die tolle Zeit fern abseits der Arbeit und dem engen Zusammenhalt auch nach der Schul- bzw. Studienzeit.

Mein aller größter Dank gilt meiner Freundin Birte. Mit ihrem Rückhalt und ihrer Geduld half sie mir auch in stressigen Phasen und unterstütze mich bis von Anfang bis Ende der Arbeit.

Karlsruhe, 06.04.2011

Asarnusch Rashid

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ACM	Association for Computing Machinery
AG	Auftraggeber
AN	Auftragnehmer
AM	Anforderungsmanagement
BBBank	Badische Beamtenbank
BMBF	Bundesministerium für Bildung und Forschung
BSWC	Basic Support for Cooperative Work
ca.	circa (Latein): etwa
bzw.	beziehungsweise
CD	Contextual Design
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
CollaBaWü	Forschungsprojekt CollaBaWü – Kollaborative, komponentenorientierte, semantikbasierte Software-Entwicklung in der Finanzbrache in Baden-Württemberg
CoP	Communities of Practice
CSCW	Computer-Supported Cooperative Work
CSD	Collaborative Software Development
CSV	Comma Separated Values (Englisch): Komma-getrennte Werte
d.h.	das heißt
DIN	Deutsches Institut für Normung e.V.
DPD	Distributed Participatory Design
EM	Ethnografische Methoden
EN	European Norm
ER-Diagramm	Entity-Relationship-Diagramme (UML)
et al.	et alii (Latein), et altera (Latein): und andere
EUD	End-User Development

f.	folgend
ff.	fortfolgend
FZI	FZI Forschungszentrum Informatik
GI	Deutsche Gesellschaft für Informatik
GQM	Goal Question Metric
GUI	Graphical User Interface (Englisch): Grafische Benutzeroberfläche
HCD	Human-Centred Design
HCI	Human-Computer Interaction (Englisch): Mensch-Maschine-Interaktion
Hrsg.	Herausgeber
HTML	HyperText Markup Language
IEEE	Institute of Electrical and Electronics Engineers
IPMA	International Project Management Association
IT	Informationstechnologie
ITK	Informations- und Telekommunikationstechnologien
IEC	International Electrotechnical Commission
IS	Information Systems (Englisch): Informationssysteme
ISD	Information System Development
ISO	International Organization for Standardization
jw.	jeweils
l.	links
LBBW	Landesbank Baden-Württemberg
MCI	Mensch-Computer-Interaktion
MIS	Management Information System
MPEG	Moving Picture Experts Group
MVC	Model-View-Controller (Entwurfsmuster)
ODS	Open-Design-Spaces
OSS	Open-Source-Software
PaDU	Participatory Design in Use

PD	Participatory Design
PDL	Program Description Language
PM	Projektmanagement
PMI	Project Management Institute
PRIMIUM	Forschungsverbund PRIMIUM – Prozessinnovationen mit Unternehmenssoftware
PSA	Problem Statement Analyzer
PSL	Problem Statement Language
r.	rechts
RSL	Requirements Statement Language
RUP	Rational Unified Process
RE	Requirements Engineering
S.	Seite
SADT	Structured Analysis and Design Technique
SE	Software Engineering
SPICE	Software Process Improvement Capability Determination (ISO 15504)
sog.	sogenannte
SyRS	System Requirements Specification
SW	Software
u.a.	unter anderem
UCD	User-Centred Design
UCSD	User-Centred System Design
UE	Usability Engineering
UI	User Involvement
UT	Usability Testing
VModell XT	VModell eXtreme Tailoring
XML	Extensible Markup Language
XP	eXtreme Programming
z.B.	zum Beispiel

1 Einleitung

Software ist heutzutage allgegenwärtig und aus dem betrieblichen Umfeld nicht wegzudenken. Es gibt kaum einen Arbeitsbereich, in dem Software nicht als Werkzeugunterstützung eingesetzt wird. Sie findet ihren Einsatz sowohl für primäre Wertschöpfungsaktivitäten als auch für unterstützende Aktivitäten in Unternehmen und trägt entscheidend zur Produktivität eines Unternehmens bei. Die Produktivitätssteigerung einer Software drückt sich u.a. dadurch aus, wie sie Mitarbeiter in ihren Arbeitsprozessen in Unternehmen unterstützt und ihre Arbeit erleichtert bzw. beschleunigt.

Das Maß für die Qualität von Software aus der Perspektive ihrer Anwender bzw. Benutzer findet sich im Terminus der Gebrauchstauglichkeit wieder. Die Gebrauchstauglichkeit beschreibt nach DIN [2004, S. 94])

> *„das Ausmaß, in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen".*

Bei der Entwicklung von Software sollte demnach der Nutzungskontext der Anwender berücksichtigt und die Steigerung der Effektivität und Effizienz nicht nur auf die reine Benutzung der Software sondern auf den gesamten umfassenden Prozess bezogen werden. Darüber hinaus sollte dem Benutzer die Benutzung so angenehm wie möglich gemacht werden, um seine Zufriedenheit sicherzustellen. Für die Gestaltung von Software reicht es demnach nicht mehr aus, dass Software gut zu bedienen ist, sondern dass Software auch bestmöglich an den Nutzungskontext der Anwender angepasst ist und eine produktive Prozessunterstützung der Arbeitsabläufe ermöglicht.

Für die Sicherstellung der Gebrauchstauglichkeit einer Software wird die Beteiligung der Anwender in Software-Projekten als entscheidender Erfolgsfaktor propagiert. Zahlreiche Untersuchungen konnten nachweisen, dass sich eine Anwenderbeteiligung positiv auf die Gebrauchstauglichkeit von Software auswirkt (vgl. [Barki und Hartwick, 1991; Baroudi et al., 1986; Brau und Schulze, 2004; Foster und Franz, 1999; Grudin, 1991a; Kujala, 2003; Kujala, 2008; McKeen und Guimaraes, 1997; Wilson et al., 1997]). In einer umfassenden Metaanalyse zu Vor- und Nachteilen der Anwenderbeteiligung stellte Kujala [2003] fest, dass eine partizipative Vorgehensweise zahlreiche positive Effekte aber auch Nachteile besitzen kann. Die Anwenderbeteiligung kann demnach die Produktivität der Anwender steigern, die Entwicklungskosten senken und den Schulungs- und Supportaufwand verringern. Außerdem wird die Akzeptanz bei der späteren Nutzung des zukünftigen Systems erhöht. Dies ist auf exaktere und fehlerfreie Anforderungen, das erhöhte Verständnis der Anwender für das System und die Steigerung der Demokratisierung der Entscheidungsfindung in Unternehmen zurückzuführen (vgl. [Kujala, 2008]).

Jedoch bringt eine Anwenderbeteiligung auch Nachteile bzw. Mehraufwände mit sich, die u.a. in drei Studien (vgl. [Hawk und Dos Santos, 1991; Heinbokel et al., 1996; Wilson et al., 1996]) beschrieben werden. Ein negativer Effekt auf den Entwicklungsaufwand ergibt sich, wenn Anwender in der finalen Phase der Entwicklung umfassende Änderungen einfordern oder wenn Moderatoren bzw. Entwickler Konflikte zwischen mehreren Anwendern bzw. Anwendergruppen lösen müssen. Insgesamt ist der Aufwand sowohl für Anwender als auch Entwickler als hoch zu bewerten, auch wenn dieser sich

positiv auf den gesamten Entwicklungsprozess auswirken kann. Des Weiteren haben Entwickler und Anwender häufig Schwierigkeiten bei der Kommunikation miteinander. Vor allem Anwender sollten über die Bedeutung und den Inhalt von Entwicklungsprozessen aufgeklärt werden, um sie produktiv einbinden zu können.

1.1 Arbeitshypothese

In Abbildung 1.1 stellt Kujala [2003; 2008] die Auswirkungen einer frühen Anwenderbeteiligung in Beziehung zur Effizienz der Produktentwicklung, der Qualität der Anforderungen und des Systems sowie der Kunden- und Anwenderzufriedenheit dar. Dass eine frühe Anwenderbeteiligung die Qualität der Anforderung und somit die Qualität des zu entwickelnden Systems steigert, ist nach ihrer Literaturanalyse eindeutig. Daraus folgt auch eine Steigerung der Zufriedenheit von Anwendern und Kunden. Allerdings lässt sie die Frage offen, wie sich eine frühe Anwenderbeteiligung auf die Effizienz der Entwicklungsteams auswirken kann. In ihren späteren Arbeiten konnte Kujala [2008] in Feldstudien Tendenzen feststellen, dass eine frühe Anwenderbeteiligung positive Effekte auf die Effizienz der Entwicklerteams haben kann.

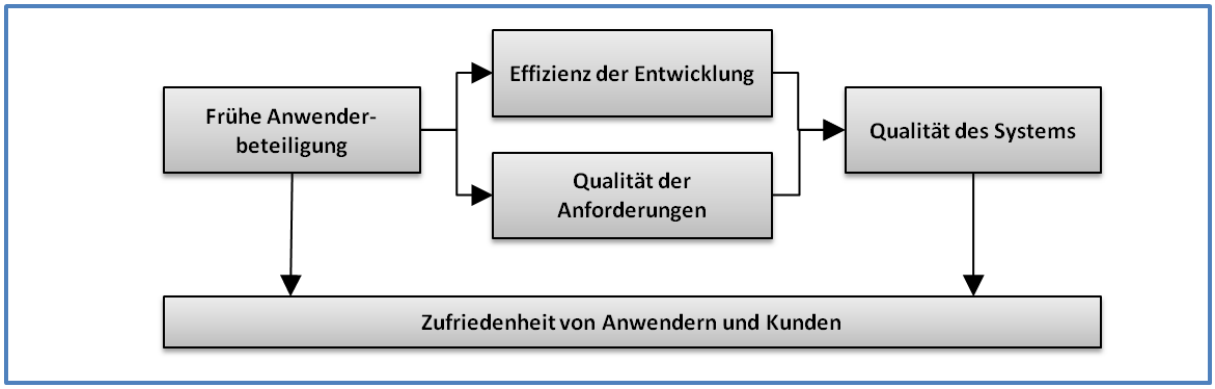

Abbildung 1.1: Auswirkung von früher Anwenderbeteiligung auf Software-Projekte [Kujala, 2008]

Kujala [2003; 2008] kommt mit ihren Untersuchungen zu der Schlussfolgerung, dass Entwickler die Anwenderbeteiligung als Chance sehen und dabei auf eine umsichtige und aktive Vorgehensweise achten sollen. Ungeeignete Maßnahmen können zu einer steigenden Anzahl von Entwicklungsiterationen, höheren Projektkosten, langsameren Entscheidungen und einer Verlängerung der Projektlaufzeit führen.

Die vorliegende Arbeit greift ihre Schlussfolgerung auf und befasst sich mit der Eignung von Maßnahmen bzw. Methoden zur Anwenderbeteiligung. Der Arbeit liegt dabei die Hypothese zugrunde, dass eine computergestützte asynchrone Kommunikation zwischen Entwickler und Anwender Vorteile gegenüber einer synchronen Kommunikation besitzt und somit ein Bedarf an der Untersuchung von Methoden mit asynchroner Kommunikation existiert.

Als synchrone Kommunikation (vgl. [Gross et al., 2007]) wird eine gleichzeitige Kommunikation zwischen Sender und Empfänger wie z.B. bei einem Gespräch oder Chat verstanden, bei der beide gleichzeitig zugegen sein müssen. Bei einer asynchronen Kommunikation zwischen Menschen erfolgt das Senden und Empfangen von Nachrichten hingegen zeitlich versetzt. Weder Sender noch Empfänger werden durch diesen Kommunikationsmodus blockiert, da der Sender nicht auf eine Antwort des

Empfängers zu warten braucht und der Empfänger nicht während des Sendens erreichbar sein muss. Bekannteste Beispiele hierfür sind Email und Diskussionsforen (vgl. [Gross et al., 2007]).

Eine Anwenderbeteiligung mit einer synchronen Kommunikation zwischen Anwendern und Entwicklern wird im Weiteren als synchrone Anwenderbeteiligung bezeichnet. Eine asynchrone Anwenderbeteiligung findet im Gegensatz dazu mithilfe einer asynchronen Kommunikation zwischen Anwendern und Entwicklern statt. Sie zeichnet sich demnach dadurch aus, dass der Anwender seinen Beitrag zu einem zur Anfrage der Entwickler zeitlich versetzen Zeitpunkt leisten kann. Bei einer asynchronen Kommunikation können sich Anwender somit zu einem vom Entwickler unabhängigen Zeitpunkt in den Prozess der Software-Entwicklung einbringen.

Aus den Erfahrungen mit asynchroner Kommunikation im Remote Usability Test (vgl. [Andreasen et al., 2007]) lassen sich mehrere mögliche Vorteile einer asynchronen Kommunikation in der Anwenderbeteiligung ableiten:

- Anwender können bei einer asynchronen Kommunikation ihre Meinung dann einbringen, wenn ihnen etwas auffällt bzw. einfällt. So können sich Anwender in aller Ruhe bzw. zu einem passenden Zeitpunkt Gedanken machen. Dabei können sie in ihrem realen Nutzungskontext bleiben und auf Probleme bzw. Lösungen in ihrem Arbeitsprozess achten (vgl. [Andreasen et al., 2007; Hartson et al., 1996; Hartson und Castillo, 1998]).

- Die Teilnahme der Anwender kann mit einer asynchronen Kommunikation vereinfacht werden, da keine festen Termine vorausgesetzt werden. Die Anwender können somit orts- und zeitunabhängig teilnehmen. Zum einen könnten so mehr Anwender beteiligt werden, zum anderen könnten Anwender ihre Teilnahmezeiten an ihrer terminlichen Auslastung ausrichten und sich mehr Zeit nehmen (vgl. [Andreasen et al., 2007; Burger et al., 2008; Hartson et al., 1996]).

- Bei einer asynchronen Kommunikation überträgt der Anwender seine Nachrichten entweder als Sprach- bzw. Videoaufzeichnung oder als schriftliches Dokument. Für den Entwickler gibt sich daraus der Vorteil, dass er den Beitrag des Anwenders in einer persistenten Form erhält.

- Anwender können auch nach Einführung einer Software und somit nach Abschluss eines Software-Projektes Verbesserungsvorschläge und Fehler melden, und somit einen kontinuierlichen Verbesserungsprozess ohne aktive Betreuung der Anwender initiieren (vgl. [Hartson et al., 1996; Hartson und Castillo, 1998]).

Mit der Anwendung von asynchroner Kommunikation in Software-Projekten sind jedoch auch mehrere Herausforderungen verbunden. Die Ergebnisse der bisherigen Evaluationen von Asynchronous Remote Usability Tests sind nach Andreasen et al. [2007] ernüchternd. Während synchrone Methoden eine hohe Akzeptanz und Erfolgsquote aufweisen, führten asynchrone Methoden zu minderwertigen Ergebnisse. Dies ist u.a. zurückzuführen auf eine fehlende Aufmerksamkeits- und Motivationssteuerung der Anwender, den höheren Aufwand der schriftlichen Kommunikation für den Anwender und den Bedarf zur Einarbeitung in das dafür erforderliche Kommunikationssystem. Somit ist vor allem bei den Anwendern ein Akzeptanzproblem vorzufinden, da sich bei asynchronen Methoden der Aufwand zur Dokumentation vom Moderator weg zum Anwender hin verschiebt.

Folglich birgt eine asynchrone Anwenderbeteiligung sowohl Chancen als auch Risiken für ein Software-Projekt, was weitere Untersuchungen erfordert. Nach Andreasen et al. [2007, S. 1413] besteht vor allem die Notwendigkeit, weitere Experimente unter Laborbedingungen und Feldstudien im realen Anwendungsfeld durchzuführen:

> *"In order to move further in that direction, it would be useful with more studies of and experiments with the asynchronous methods. In addition, it would be interesting to perform comparative studies of remote usability testing methods outside the controlled environment of a usability laboratory."*

Es fehlt demnach an weiteren Studien und Experimenten außerhalb von kontrollierten Laborumgebungen, um die bisherigen Erfahrungen und Erkenntnisse überprüfen sowie neue Erkenntnisse über die Potentiale einer asynchronen Anwenderbeteiligung gewinnen zu können.

1.2 Ziele der Arbeit

Ziel dieser Arbeit ist es, die Potentiale von asynchroner Anwenderbeteiligung in Software-Projekten zu untersuchen. Die Arbeit adressiert folgende Fragestellungen:

1. Welche Bewertungskriterien sind für Methoden zur Anwenderbeteiligung anzuwenden?

   Für die Bewertung von Methoden zur Anwenderbeteiligung sind Bewertungskriterien zu definieren, die darüber Aufschluss geben, wie Effizienz und Effektivität einer Anwenderbeteiligung operationalisiert und schließlich Aussagen über die Effizienz des Entwicklungsteams getroffen werden können. Während sich die Effizienz einer Anwenderbeteiligung durch den zeitlichen, materiellen und kognitiven Aufwand beziffern lässt, drückt sich die Effektivität durch die Anzahl und Qualität der Anwenderbeiträge aus. Ferner ist auch die subjektive Zufriedenheit bzw. Belastung der Anwender zu berücksichtigen, um die Akzeptanz einer Methode feststellen zu können.

2. Was leisten Methoden zur synchronen Anwenderbeteiligung und wo liegen ihre Grenzen?

   Das Spektrum an etablierten Methoden zur Anwenderbeteiligung besteht vorwiegend aus Methoden mit synchroner Kommunikation, wie z.B. Interviews, Workshops, Beobachtungen, etc. Dementsprechend liegen zahlreiche Untersuchungen und Erfahrungen über ihre Vor- und Nachteile vor, deren Analyse Aufschluss darüber geben kann, ob und wo Bedarf an Methoden mit asynchroner Kommunikation besteht.

3. Welchen Nutzen kann eine asynchrone Anwenderbeteiligung stiften?

   Wie eingangs motiviert, sind bei einer asynchronen Anwenderbeteiligung Vor- und Nachteile zu erwarten, die im Rahmen dieser Arbeit im Detail untersucht werden sollen. Dabei stellt sich die Frage, welche Eigenschaften eine asynchrone Anwenderbeteiligung besitzt und wie sich diese auf Software-Projekte auswirken können.

4. Kann mit einer neuen Methode zur asynchronen Anwenderbeteiligung die Effizienz der Entwicklung, die Qualität der Anwenderbeiträge und die Belastung der Anwender verbessert werden?

Die Art und Weise, wie eine asynchrone Anwenderbeteiligung umgesetzt wird, kann sich auf die Effizienz der Entwicklungsteams, auf die Qualität der Anwenderbeiträge und auf die Belastung der Anwender auswirken. Daher soll aufbauend auf einer Analyse der existierenden Methoden exemplarisch eine neue Methode entwickelt und ihre Effekte auf die Effizienz der Entwicklung, auf die Qualität der Anwenderbeiträge und auf die Belastung der Anwender untersucht werden.

Daher ist zu Beginn zu klären, welche Bewertungskriterien an Methoden zur Anwenderbeteiligung angewendet werden können. Anschließend sind Potentiale und Grenzen von Methoden zur synchronen Anwenderbeteiligung zu untersuchen und Verbesserungsmöglichkeiten durch Methoden zur asynchronen Anwenderbeteiligung aufzuzeigen. Schließlich ist exemplarisch zu demonstrieren, wie eine Methode zur asynchronen Anwenderbeteiligung zu gestalten ist und wie sie sich auf Software-Projekte auswirkt. Dabei soll nicht ausgeschlossen werden, dass in einem Software-Projekt auch eine Kombination von synchroner und asynchroner Anwenderbeteiligung sinnvoll sein kann.

1.3 Aufbau der Arbeit

Die Arbeit ist in Anlehnung an die vier Forschungsfragen aus Kapitel 1.2 in vier Teile und zehn Kapitel gegliedert.

Teil 1 umfasst das Kapitel 2 und befasst sich mit der ersten Forschungsfrage nach Bewertungskriterien für Methoden der Anwenderbeteiligung.

In Kapitel 2 werden zunächst Charakteristika und grundlegende Rahmenbedingungen der Anwenderbeteiligung beschrieben. U.a. wird aufgearbeitet, welche Aktivitäten eine Anwenderbeteiligung umfasst, welche Dimensionen der Anwenderbeteiligung existieren, welche Rollen beteiligt sind und an welche Phasen und Prozessartefakten eines Software-Projektes eine Anwenderbeteiligung anknüpfen kann. Darauf aufbauend werden die Bedeutung der Anwenderbeteiligung für Software-Projekte erläutert und der aktuelle Stand der Forschung zur Thematik sowie die etablierten Ansätze vorgestellt. Schließlich werden Vor- und Nachteile der Anwenderbeteiligung diskutiert und daran anlehnend Bewertungskriterien für Methoden der Anwenderbeteiligung abgeleitet.

In Teil 2 wird mit dem Kapitel 3 die zweite Forschungsfrage nach den Eigenschaften von synchroner Anwenderbeteiligung und ihren Grenzen behandelt.

Das dritte Kapitel gibt zu Beginn einen Überblick über Methoden zur synchronen Anwenderbeteiligung. Mithilfe der in Kapitel 2 entwickelten Bewertungskriterien findet zu jeder Methode eine Diskussion über ihre Vor- und Nachteile statt. Die Diskussion mündet schließlich in eine Zusammenfassung der Defizite der vorhandenen Methoden und die Motivation der Notwendigkeit von Methoden zur asynchronen Anwenderbeteiligung.

Teil 3 bezieht sich mit dem Kapitel 4 auf die dritte Forschungsfrage und diskutiert die Eigenschaften asynchroner Anwenderbeteiligung und deren Potentiale zur Verbesserung von Software-Projekten.

Das vierte Kapitel stellt existierende Methoden zur asynchronen Anwenderbeteiligung vor und fasst die bisherigen Erkenntnisse zu ihren Vor- und Nachteilen zusammen. Schließlich werden die Potentiale einer asynchronen Anwenderbeteiligung resümiert und Defizite der existierenden Methoden

identifiziert. Dabei wird auch die Frage nach Möglichkeiten zur Vereinfachung der schriftlichen Formulierung von Anwenderbeiträgen aufgeworfen und die Idee für den Einsatz von Annotationssystemen motiviert.

Kapitel 5 befasst sich mit den Potentialen von Annotationssystemen zur Verbesserung der Kommunikation zwischen Entwickler und Anwender. Nach einer motivierenden Einführung in die Grundlagen zu Annotationen und Annotationssystemen werden existierende Ansätze vorgestellt und die bisherigen Erfahrungen zusammengefasst. Dem schließt sich eine Diskussion über Anwendungsmöglichkeiten von Annotationssystemen zur asynchronen Anwenderbeteiligung an.

Im vierten Teil wird mit den Kapiteln 6, 7, 8 und 9 der vierten Forschungsfrage nach den Verbesserungsmöglichkeiten durch Methoden zur asynchronen Anwenderbeteiligung nachgegangen. Um ihre Potentiale auszuschöpfen und ihre Defizite zu beheben, wird eine neue Methode entwickelt und im Rahmen einer Evaluation mit anderen Methoden zur synchronen und asynchronen Anwenderbeteiligung verglichen.

Im sechsten Kapitel erfolgt basierend auf den in dieser Arbeit gewonnenen Erkenntnissen die Konzeption einer neuen Methode zur asynchronen Anwenderbeteiligung unter der Bezeichnung OpenProposal. In Anlehnung an die Erfolgsfaktoren zur Anwenderbeteiligung und die Bewertungskriterien an Methoden zur Anwenderbeteiligung werden Ziele, Lösungskonzepte, Anforderungen und Prozessmodell von OpenProposal bestimmt.

Das siebte Kapitel beschreibt die Implementierung des OpenProposal Annotationssystems. Dabei werden die Systemarchitektur, das Datenmodell, die Benutzeroberfläche und das Bedienkonzept von OpenProposal berücksichtigt.

Kapitel 8 beschreibt die Evaluation der Gebrauchstauglichkeit des OpenProposal Annotationssystem. Hierfür wird OpenProposal im Rahmen von Usability Tests mit 36 Testpersonen getestet und die Verbesserungsmöglichkeiten sowie das Konzept der Annotation von Bildschirmfotos werden zur Diskussion gestellt. Die Ergebnisse der Tests lassen schlussfolgern, dass Anwender den Ansatz von OpenProposal begrüßen und das Annotationssystem für sinnvoll befinden.

In Kapitel 9 erfolgt die Evaluation der OpenProposal Methode im Rahmen von zwei Fallstudien, um die Effekte der asynchronen Anwenderbeteiligung im anwendungsnahen Umfeld zu untersuchen. Zu Beginn werden die Grundlagen zur Gestaltung, Durchführung und Auswertung von Fallstudien zusammengefasst. Dabei wird motiviert, dass die Methode der vergleichenden Fallstudien für die Evaluation von OpenProposal als geeignet einzustufen ist. Demzufolge wird zuerst die Planung der vergleichenden Fallstudien vorgenommen, dann werden die beiden Fallstudien durchgeführt und ausgewertet, und im Anschluss eine fallübergreifende Diskussion geführt. Zu jeder Fallstudie werden die Beteiligten, die Historie, der Verlauf, die Auswertung der Beobachtungen und Befragungen sowie die Schlussfolgerungen präsentiert. Die erste Fallstudie umfasst die Einführung von OpenProposal bei dem Unternehmen TRUMPF. Im Rahmen eines Usability Workshops wird OpenProposal eingesetzt, bewertet und mit anderen Ansätzen verglichen. In der zweiten Fallstudie setzt das Team des Forschungsprojekts WAVES OpenProposal ein. Über mehrere Monate werden Anwender mithilfe von OpenProposal und anderen Methoden an der Entwicklung von Forschungsprototypen beteiligt. Ergebnis der fallübergreifenden Analyse ist, dass OpenProposal positiv hervorsticht und einen Nutzen für alle Beteiligten stiften konnte. U.a. konnte gezeigt werden, dass die Anwender im Vergleich zu

anderen Methoden mit OpenProposal eine höhere Anzahl an Beiträgen erstellten, die Qualität der Beiträge höher und die Zufriedenheit der Anwender größer war.

Im letzten Kapitel 10 werden die Ergebnisse der Arbeit zusammengefasst und diskutiert, die Grenzen des Forschungsansatzes erörtert, der methodische Ansatz kritisch hinterfragt sowie ein Ausblick auf technische Erweiterungspotentiale und mögliche weitere Forschungsarbeiten gegeben.

2 Bewertungskriterien für Methoden zur Anwenderbeteiligung

Ziel dieses Kapitel ist es, Bewertungskriterien für Methoden der Anwenderbeteiligung zu erarbeiten. Hierfür wird zu Beginn ein Überblick über den theoretischen Überbau zu Anwenderbeteiligung in Software-Projekten und die für die Arbeit relevanten theoretischen Grundlagen gegeben. Darauf aufbauend wird der Stand der Forschung zur Anwenderbeteiligung zusammengefasst und es werden Kriterien für die Evaluation von Methoden zur Anwenderbeteiligung extrahiert. Ergänzend bringen Experten im Rahmen von Interviews ihre Erfahrungen aus der Praxis ein und erläutern aus ihrer Sicht die Erwartungen an eine Anwenderbeteiligung in Software-Projekten. Anhand der Ergebnisse aus der Literaturanalyse und den Experteninterviews werden Risiken, Erfolgsfaktoren und Qualitätsmerkmale abgeleitet, die zur Definition von Bewertungskriterien für Methoden der Anwenderbeteiligung führen.

2.1 Software und Software-Projekt

Software ist nach Balzert [2000a] in Anlehnung an die Brockhaus Enzyklopädie [2003] „*die Gesamtheit oder Teil der Programme für Rechensysteme, wobei Programme zusammen mit den Eigenschaften der Rechensysteme, die Nutzung der Rechensysteme zur Lösung gestellter Aufgaben oder zusätzlicher Betriebs- und Anwendungsarten der Rechnersysteme ermöglichen*“.

Im Zusammenhang mit dem Begriff Software tauchen auch die Begriffe Software-Produkt und Software-System auf. Ein Software-Produkt bezeichnet nach Balzert [2000a] eine spezifische Sicht des Käufers bzw. Auftraggeber von außen auf Software, während ein Software-System die innere Sicht des Entwicklers auf eine Software ist, bei der eine Software in Systemkomponenten unterteilt ist. Außerdem wird bei Software zwischen Systemsoftware und Anwendungssoftware unterschieden. Systemsoftware ist Software, die für eine spezielle Hardware oder eine Hardwarefamilie entwickelt wurde, die den Betrieb dieser Hardware ermöglicht bzw. erleichtert. Dazu gehören u.a. Betriebssysteme, Compiler, Datenbanken und Kommunikationsprogramme. Anwendungssoftware ist Software, die Aufgaben des Anwenders mit Hilfe eines Computersystems löst. Anwendungssoftware setzt in der Regel auf der Systemsoftware der verwendeten Hardware auf. Anwendungssoftware, Systemsoftware und Hardware bilden zusammen ein Computer- bzw. DV-System. Die ausschließliche Entwicklung einer Software bezeichnet man als Software-Entwicklung, wo hingegen eine System-Entwicklung die Entwicklung eines Systems ist, das Software- und Hardwarekomponenten umfasst (vgl. [Balzert, 2000a]). Da in der Literatur und im gängigen Sprachgebrauch die Begriffe System und Software häufig synonym verwendet werden und viele Erkenntnisse aus dem Systems Engineering auch für Software Engineering gültig sind, soll in dieser Arbeit zwischen den beiden Begriffen Software-System und Software nicht explizit unterschieden werden.

Unter einem Projekt versteht man gemäß der International Project Management Association[2] (IPMA) eine einmalige Gesamtheit von koordinierten Aktivitäten mit bestimmten Anfangs- und Endpunkten, die von einer Person oder Organisation mit dem Ziel durchgeführt werden, bestimmte Termin-, Kosten- und Leistungsziele zu erreichen (vgl. [Balzert, 2000a]). Die wesentlichen Merkmale von Projekten sind nach Stahlknecht und Hasenkamp [2005] die Einmaligkeit des Unternehmens (im Gegensatz zu Geschäftsprozessen), die Zusammensetzung aus Teilaktivitäten, die Beteiligung von Personen und/oder Stellen (Rollen) unterschiedlicher Fachrichtungen (Interdisziplinarität), Teamarbeit (Kollaboration), das Konkurrieren mit anderen Projekten um Personal- und Sachmittel, Mindest- und Höchstwerte für Dauer und Aufwand (Budget und Zeitrahmen) sowie ein definierter Anfang und ein definiertes Ende (Projektziel).

Ein Software-Projekt ist demnach ein Projekt mit dem Ziel der Entwicklung, Anpassung bzw. Wartung einer Software. Sommerville [1996, S. 5] kategorisiert Software-Projekte nach ihren Zielgruppen und unterscheidet zwischen generischen Produkten (*„generic products“*) , die als Massenprodukt an jeden Interessierten verkauft werden, und individuellen Lösungen (*„bespoke custimized products*“), die für einen Kunden speziell entwickelt werden. Grudin [1991a; 1996] differenziert die individuellen Lösungen zusätzlich in Auftragsentwicklung („contract development“) und interne Entwicklung („in-house development and custom development“). Die drei Kategorien unterscheiden sich nach Grudin [1996] stark in ihrem Entwicklungskontext:

- Produktentwicklung: Zielgruppe dieser Form der Entwicklung, auch als „Off-the-Shelf product development“, „Entwicklung für den Markt“ oder „consumer products“ bezeichnet, ist der Massenmarkt. Die Anforderungen an die Software werden vom Hersteller intern, und hier meist von Marketing und Vertrieb, definiert. Das Vorgehensmodell kann vom Hersteller nach seinen individuellen Vorstellungen gewählt werden. Die Kunden und Anwender können durch ihre Kaufentscheidung und ihre Rückmeldungen zum Produkt indirekt Einfluss auf die Entwicklung nehmen. Die Berücksichtigung individueller Wünsche einzelner Kunden oder Anwender findet nicht statt. Zeit- und Kostendruck sind sehr hoch.

- Auftragsentwicklung: Ein großer Anteil von Software-Projekten erfolgt gezielt durch einen Auftrag einer oder mehrerer Organisationen an ein Software-Unternehmen. Die Anforderungen werden in Form eines formalen Vertrags zwischen Kunde und Entwickler festgehalten. Somit werden Anforderungen an die geplante Software zu großen Teilen vor Beginn der Entwicklung definiert. Das Vorgehensmodell wird zwischen Kunde und Entwickler individuell abgestimmt und ist ebenfalls Gegenstand des Vertrags. Der Kunde kann direkt auf die Entwicklung Einfluss nehmen. Der Anwender kann im Laufe des Projektes zur Konkretisierung von Anforderungen eingebunden werden. Zeit- und Kostendruck sind auch bei dieser Form der Entwicklung sehr hoch.

- Interne Entwicklung: Bei internen Software-Projekten arbeiten Entwickler und Anwender in der gleichen Organisation. Externe Entwickler werden bei Bedarf hinzugezogen. Das Vorgehensmodell kann vom Entwickler intern definiert und mit den externen Entwicklern abgestimmt werden. Die Entwicklung erfolgt meist inkrementell. Die Anforderungen werden im

[2] http://www.ipma.ch/, abgerufen am 17.05.2010.

Laufe der Entwicklung an die Bedürfnisse der Anwender angepasst. Wird die Software in Betrieb genommen, kann der Anwender weiterhin Kontakt mit den Entwicklern aufnehmen und weitere Vorschläge einbringen. Eine vertragliche Fixierung des Projekts existiert nicht. Zeit- und Kostendruck ist im Regelfall gering.

Ferner können nach Grudin [1991a] interne und externe Faktoren Einfluss auf ein Software-Projekt nehmen. Interne Faktoren sind u.a. die Größe, die Prinzipien und Paradigmen sowie die organisatorischen Strukturen der beteiligten Organisation. Außerdem ist ein Projekt von den Eigenschaften der beteiligten Anwendergruppen und anderen beteiligten Partnern im Projekt abhängig. Zudem spielt der Grad der Innovation bzw. Unsicherheit der geplanten Software eine entscheidende Rolle. Ebenfalls von Bedeutung sind das Engagement der Projektbeteiligten und die Form der vertraglichen Vereinbarungen. Als externe Faktoren können sich Veränderungen auf dem Arbeitsmarkt, Wettbewerbsdruck, Technologiefortschritt, Standards, gesetzliche Regulierungen, etc. auf ein Projekt auswirken.

Software-Projekte sind somit von vielen Faktoren und großer Unsicherheit geprägt. Diesen Unsicherheiten zu begegnen ist Aufgabe einer ingenieurmäßigen Vorgehensweise in Software-Projekten, der so genannten Software-Technik („Software Engineering“). Die Software-Technik als Teildisziplin der Informatik befasst sich mit der Herstellung und Anwendung von Software. Sie versteht sich nach Balzert [2000a, S. 36] als die *„zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden, Konzepten, Notationen und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Software-Systemen“*. Als Prinzipien werden abstrakte Grundsätze verstanden, die man seinem Handeln zugrunde legt. Sie werden aus der Erfahrung und Erkenntnis hergeleitet bzw. durch sie bestätigt. Methoden sind planmäßig angewandte, begründete Vorgehensweisen zur Erreichung von festgelegten Zielen und orientieren sich im Allgemeinen an festgelegten Prinzipien. Verfahren werden von Balzert [2000a] synonym zu Methoden verwendet. Konzipieren bedeutet, „eine Grundvorstellung von etwas entwickeln, verfassen, entwerfen“ [Duden, 2007]. In Konzepten werden somit definierte Sachverhalte unter bestimmten Aspekten modelliert. Ein Konzept nach Balzert [2000a] lässt sich auch synonym mit Modell bezeichnet (vgl. [Mertens et al., 2001]). Zur Beschreibung von Konzepten bzw. Modellen werden Notationen verwendet, bei denen Informationen durch Symbole dargestellt werden. Werkzeuge dienen der automatisierten Unterstützung und Einhaltung von Methoden, Verfahren, Konzepten und Notationen.

2.2 Phasen in Software-Projekten

Die Software-Entwicklung besteht aus einer Vielzahl von teambasierten Aktivitäten, deren Ergebnisse einzelne Teilprodukte sind. Solche Aktivitäten werden nach zeitlichen, begrifflichen, technischen und organisatorischen Kriterien geordnet. Die Bezeichnung dieser Phasen wird zwar in der Literatur unterschiedlich gehandhabt, aber prinzipiell in ähnlicher Form verwendet. Die Definition dieser Aktivitäten basiert ursprünglich auf Canning [1956] und wurde im Laufe der Jahrzehnte weiterentwickelt und aus unterschiedlichen Blickwinkeln diskutiert (vgl. [Agresti, 1986; Balzert, 2000a; Bennington, 1956; Boehm, 1988; Laden und Gildersleeve, 1963; Royce, 1970; Royce, 1986; Sommerville, 1996]). Dabei steht zentral die Forderung einer ingenieurmäßigen systematischen Vorgehensweise in der Software-Entwicklung im Vordergrund. Generell durchläuft ein Software-Projekt die Phasen „Anforderungsana-

lyse“, „Entwurf und Implementierung“, „Test“, „Abnahme und Einführung“, sowie „Wartung und Pflege“. Außerdem existieren weitere parallel verlaufende Querschnittsaktivitäten u.a. „Projektmanagement“, „Konfigurations- und Änderungsmanagement“ und „Qualitätsmanagement“.

Die erste Phase der Software-Entwicklung, die Anforderungsanalyse, wird oft auch nur Analyse (vgl. [Stahlknecht und Hasenkamp, 2005]), Systemanalyse (vgl. [Noack und Schienmann, 1999]), Definitionsphase (vgl. [Balzert, 2000a]) oder Software Requirements-Phase (vgl. [Sommerville, 1996]) genannt. Ziel der Anforderungsanalyse ist die Identifikation, die Beschreibung und die Qualitätssicherung von Anforderungen. Die Anforderungsanalyse lässt sich in die vier Phasen „Machbarkeitsstudie“, „Anforderungserhebung und –analyse“, „Anforderungsspezifikation“ und „Anforderungsvalidierung“ unterteilen. Zudem sieht Sommerville [1996] das Anforderungsmanagement als zusätzliche, nebenläufige Aktivität, die sich mit der Verwaltung von Anforderungsänderungen beschäftigt. Anforderungen selbst können anhand ihrer Abstraktionsebene (Benutzeranforderungen vs. Systemanforderungen) sowie ihres Ursprungs (funktionale vs. nichtfunktionale Anforderungen) unterschieden werden. Stahlknecht und Hasenkamp [2005] unterscheiden zudem in der Analysephase zwischen der „Istanalyse“ und der Entwicklung von „Sollkonzepten“, was die Anforderungsanalyse im eigentlichen Sinn darstellt. In der Literatur wird überdies häufig auch in Analogie zum Business Engineering von Requirements Engineering gesprochen (vgl. [Sommerville, 1996, S. 121ff]).

Die Ergebnisse der Definitionsphase bzw. Anforderungsanalyse bilden nach Balzert [2000a] den Ausgangspunkt bzw. die Grundlage für den Software-Entwurf. Im Hinblick auf die nachfolgende Implementierungsphase spricht er auch von der „*Programmierung im Großen*“ (vgl. [Balzert, 2000a, S. 686]), da beim Entwurf bereits die Software-Architektur mit den technischen Details festgelegt wird, zu der auch die Bestimmung der Zielplattform für die Implementierung zählt. In der englischsprachigen Literatur wird diese Phase allgemein auch häufig als Software Design bezeichnet (vgl. [Sommerville, 1996]).

In der Implementierungsphase werden alle Aktivitäten zur tatsächlichen Umsetzung des Entwurfmodells zu einem ausführbaren Anwendungssystem gebündelt. Beispiele von typischen Implementierungsaktivitäten sind die Konzeption von Datenstrukturen und Algorithmen sowie die Umsetzung der Entwurfsmodelle und -muster in die Konstrukte der eingesetzten Programmiersprache. Weitere Bezeichnungen für diese Phase sind u.a. Software Construction (vgl. [Abran und Moore, 2004]) und Realisierungsphase (vgl. [Stahlknecht und Hasenkamp, 2005]).

Das Testen von Programmteilen, Teilsystemen (Unit Tests) oder des gesamten Anwendungssystems (Integrationstest) wird häufig auch als begleitende, parallele Aktivität in der Implementierungsphase mit aufgeführt, oder aber die Tests werden als Teil der Querschnittsfunktion „Qualitätsmanagement“ oder „Qualitätssicherung“ orthogonal zum gesamten Software-Entwicklungsprozess aufgefasst (vgl. [Balzert, 2000a]). Vor der Abnahme durch den Kunden und der Einführung im Unternehmen sollte jedoch in jedem Fall eine abschließende Testphase durchlaufen werden.

In der Abnahme- und Einführungsphase wird die fertige Software erstmals als Ganzes übergeben und anschließend beim Anwenderunternehmen eingeführt, also in Betrieb genommen. Im Englischen wird dieser Prozess auch als „Deployment“ bezeichnet. Zu den Aktivitäten der Einführungsphase zählen u.a. Installation, Mitarbeiterschulung und Inbetriebnahme. Ab dem Zeitpunkt der ersten Inbe-

triebnahme befindet sich die Software im Systembetriebsmodus und geht dann in die letzte Phase im allgemeinen Software-Lebenszyklus über in die Wartung und Pflege.

Den Oberbegriff sowohl für Wartung als auch Pflege bildet nach Balzert [2000a], Abran und Moore [2004] die Software Maintenance. Da mit sehr großer Wahrscheinlichkeit im laufenden Betrieb einer Software Fehler auftreten werden und sich ändernde Umweltbedingungen und/oder Benutzeranforderungen Anpassungen erfordern, ist diese Phase wichtig, um den Software-Lebenszyklus zu verlängern und zu maximieren (vgl. [Balzert, 2000a]).

Als Projektmanagement wird laut Stahlknecht und Hasenkamp [2005, S. 219] „*die Gesamtheit aller Tätigkeiten bezeichnet, mit denen Projekte geplant, überwacht und gesteuert werden*“. Dies umfasst insbesondere „*die Gesamtheit aller Koordinations- und Führungsaufgaben, die sich auf ein Projekt beziehen. Es ist die Anwendung von Wissen, Fertigkeiten, Werkzeugen und Verfahren auf Projektvorgänge, um die Bedürfnisse und Erwartungen der Stakeholder an ein Projekt zu erfüllen oder zu übertreffen*“ (vgl. [PMI, 2003]). In der Literatur finden sich weitere Bezeichnungen für diesen Aufgabenbereich, z.B. Planungsphase (vgl. [Balzert, 2000a]), Projektbegründung und –management (vgl. [Stahlknecht und Hasenkamp, 2005]), Project Management (vgl. [Sommerville, 1996]) und Software Engineering Management (vgl. [Abran und Moore, 2004]).

Unter einem Konfigurations- und Änderungsmanagement versteht man im Allgemeinen die fortlaufende Dokumentation der Systementwicklung, insbesondere aller System- und Programmanforderungen, des aktuellen Systemstatus, der vorgenommenen Änderungen während der Systementwicklung und der neu entstehenden Programmversionen einschließlich der Unterschiede zu der jeweils vorangegangenen Version während des Systembetriebs. Bei Balzert [2000a] wird das Konfigurations- und Änderungsmanagement als Mittel aufgefasst, um eine geordnete Abwicklung der Wartungsaufgaben sicherzustellen, ist somit also Teil der Software-Lebenszyklusphase „Wartung und Pflege“.

Der Standard DIN ISO 9126 liefert für Software-Qualität die folgende Definition:

> *„Software-Qualität ist die Gesamtheit der Merkmale und Merkmalswerte eines Software-Produkts, die sich auf dessen Eignung beziehen, festgelegte oder vorausgesetzte Erfordernisse zu erfüllen“.*

2.3 Vorgehensmodelle in Software-Projekten

Abhängig von der Projektorganisation findet der Ablauf der einzelnen Phasen in Software-Projekten entweder sequentiell im Sinne des Wasserfallmodells (vgl. [Bennington, 1956; Royce, 1970; Royce, 1986]) oder dem V-Modell XT (vgl. [Broy und Rausch, 2005]), inkrementell nach dem Evolutionsmodell (vgl. [Balzert, 2000a]) oder iterativ in Form des Spiralmodells (vgl. [Boehm, 1988]) statt. Eine Spezialform der inkrementellen und iterativen Vorgehensweise stellt das Prototyping (vgl. [Budde et al., 1992; Schönthaler, 1989]) dar.

Im Wasserfallmodell verläuft die Entwicklung streng sequentiell und jede Phase beginnt erst wenn die vorherige abgeschlossen wurde. Die Anforderungen werden dabei einmalig zu Beginn eines Projektes in der Anforderungsanalyse festgelegt. Eine nachträgliche Anpassung des Entwurfes bzw. der Anforderungen sollen in späteren Phasen nicht vorgenommen werden.

Bei der inkrementellen Entwicklung wird eine Software Stück für Stück entwickelt. Dabei wird jedes Teil der Software eigenständig für sich entworfen, umgesetzt und getestet bevor es in das Gesamtsystem der Software integriert wird. Zu Beginn muss somit die Systemarchitektur einmalig festgelegt werden. Dank der schrittweisen Vorgehensweise können die Teilsysteme nachträglich angepasst werden, so dass die Anforderungen nicht zu Beginn vollständig festgelegt werden müssen, sondern auch im Laufe der Entwicklungen neue bzw. angepasste Anforderungen erfasst werden können. Da die Systemarchitektur zu Beginn ohne genaue Kenntnisse der Anforderungen festgelegt wird, ist eine nachträgliche Anpassung der Systemarchitektur mit einem hohen Zusatzaufwand verbunden, wenn sich während der Entwicklung herausstellt, dass die Architektur ungünstig gewählt wurde.

Im Spiralmodell werden die einzelnen Phasen iterativ in mehreren Zyklen nacheinander durchlaufen, wobei die Software weiter entwickelt wird. Im Gegensatz zur inkrementellen Entwicklung wird die Software nicht in Teile zerlegt sondern im Ganzen iterativ entworfen und umgesetzt. Die Anforderungen an die Software werden in jedem Zyklus überprüft und anhand der gewonnenen Erkenntnisse aus dem letzten Zyklus angepasst. In den letzten Jahren wurden zahlreiche Adaptionen der iterativen Prozessmodelle entwickelt, beispielsweise Methoden der Agilen Software-Entwicklung (vgl. [Cockburn, 2003]) und eXtreme Programming (vgl. [Beck, 1999]).

Im Prototyping wird ein früher Prototyp zur finalen Software inkrementell bzw. iterativ entwickelt. Zuerst wird im Groben ein erster Entwurf erstellt, wobei in diesem Zusammenhang erste Anforderungen erhoben werden. Basierend auf Rücksprachen mit den Anwendern können die Anforderungen angepasst und der Prototyp überarbeitet werden. Ein Prototyp kann nach Sommerville [1996, S. 141ff] dabei eine Zwischenstufe der Entwicklung darstellen oder als „Wegwerf-Prototyp“ („Throw-away-prototype“) erstellt und einmalig zum Testen der Anforderungen verwendet werden. Ein Prototyp kann nach Kieback et al. [1992] und Budde et al. [1992] entweder als Demonstrationsprototyp, als Prototyp im engeren Sinn, als Labormuster oder als Pilotsystem zum Einsatz kommen. Demonstrationsprototypen werden häufig in der Planung und Akquise eingesetzt und stellen einen ersten Entwurf der grafischen Benutzeroberfläche dar. Prototypen im engeren Sinne werden durch ein lauffähiges Provisorium abgebildet und sind hilfreich für erste frühe Tests. Labormuster stellen mehrere Alternativen dar, die in Tests untersucht werden und somit Entwurfsalternativen experimentell selektieren können. Dem am weitesten fortgeschrittenen Stadium entspricht das Pilotsystem, das die Funktionalität der geplanten Software weitgehend enthält und schließlich inkrementell zur finalen Software entwickelt wird.

2.4 Rollenmodelle in Software-Projekten

Eine Rolle ist die Beschreibung einer Menge von Aufgaben und Verantwortlichkeiten im Rahmen eines Projekts und einer Organisation. Abhängig vom Projektvorhaben werden Teams gebildet, die aus Menschen mit unterschiedlicher Ausbildung, unterschiedlichen Zielen und Methoden bestehen. Die Zuordnung von Organisationseinheiten und Personen zu den Rollen erfolgt zu Beginn eines Projekts. Dabei kann eine Person mehrere Rollen besetzen, es kann aber auch eine Rolle durch mehrere Personen besetzt werden.

In der Literatur werden - wie bereits bei den Projektphasen - unterschiedliche Bezeichnungen für die einzelnen Rollen verwendet. Grundsätzlich können nach der Literatur (vgl. [Balzert, 2000b; Macaulay,

1993; Pfleeger, 1998]) vier Rollenklassen von Projektbeteiligten gebildet werden, die direkt in der Software-Entwicklung involviert sind und eine grobe Klassifikation der Zuständigkeiten in einem Software-Projekt umreißen. Bei der ersten Klasse handelt es sich um die Entwickler, die für die Gestaltung und Umsetzung der Software verantwortlich sind. Die zweite Klasse, die der Anwender, soll mit diesem Produkt arbeiten und bei ihrer Arbeit unterstützt werden. Darüber hinaus existiert die Klasse der Entscheider, die für die finanziellen und strategischen Aspekte eines Software-Projektes zuständig ist. Die vierte Klasse, die der Moderatoren, ist für die Einführung der Software und die Schulung der Anwender verantwortlich. Diese vereinfachte Einteilung soll in dieser Arbeit die Komplexität der Rollen in Software-Projekte reduzieren und die Lesbarkeit erhöhen. Darüber hinaus existieren weitere Rollen, die nicht direkt an der Software-Entwicklung sondern u.a. in der Logistik, dem Marketing, den Schulungen etc. beteiligt sind. In Tabelle 2.1 werden die Rollendefinitionen aus einer Auswahl an bekannter Literatur vorgestellt und den vier Rollenklassen zugeordnet.

Rolle / **Quelle**	**Entwickler**	**Anwender**	**Entscheider**	**Moderator**
[Macaulay, 1993]	Software Designer	User, User Representative	Project Manager	User Manager, System Analyst
[Balzert, 2000a]	Software-Entwerfer, Implementierer	Anwender	Software-Manager, Projekt-Manager, Projekt-Leiter	Systemanalytiker, Anwendungsspezialist
[Pfleeger, 1998]	Programmer, Software Engineer, Tester, Designer	Customer, User	-	Requirements Analyst
[Broy und Rausch, 2005]	Software-Architekt, Software-Entwickler	Anwender	Projektleiter, Lenkungsausschuss, Projektmanager	Anforderungsanalytiker
[Kruchten, 2000]	Software Architect, Designer, Implementer, Integrator, User-Interface-Designer, Database Designer	User, Customer, User Representative, Buyer	Project Manager, Test Manager, Process Engineer, Change Control Manager, Configuration Manager, Management Reviewer	System Analyst, Requirements Specifier

Tabelle 2.1: Zuordnung des Rollenverständnisses aus der Literatur auf die Rollenklassen

2.4.1 Entwickler

Die Gruppe der Entwickler besteht aus unterschiedlichen Spezialisten für Entwurf und Realisierung einer Software. Als Spezialisierungsformen nennt Balzert [2000a] u.a. den Software-Architekten, Software-Entwerfer, den Software-Ergonomieverantwortlichen, den Änderungsverantwortlichen, etc. Ziel der Entwickler ist die Entwicklung der bestmöglich an die Anforderungen angepassten Soft-

ware, die mit den vorgegebenen Ressourcen und dem aktuellen Stand der Technik möglich ist. Das V-Modell XT (vgl. [Broy und Rausch, 2005]) sieht folgende Definition für den Software-Entwickler vor:

> *„Der »SW-Entwickler ist für die Realisierung der »SW-Elemente auf Basis der »SW-Spezifikation zuständig."*

Somit liegt dem Entwickler eine Spezifikation vor, an der er sich orientieren soll. Die Entwicklung erfordert neben der eigentlichen Implementierungsarbeit auch einen hohen Aufwand für Teamkommunikation und –organisation, da im Regelfall mehrere Entwickler mit unterschiedlichen Perspektiven und Verantwortlichkeiten an einem Software-Projekt beteiligt sind.

2.4.2 Anwender

Zur Gruppe der Anwender zählen die Menschen, die von der Anwendung einer Software betroffen sind. Häufig wird auch zwischen Anwender und Benutzer unterschieden. Balzert [2000a] sieht hier den Unterschied, dass Benutzer „nur diejenigen Personen [sind], die ein Computersystem unmittelbar einsetzen und bedienen", während als Anwender „alle Angehörigen einer Institution oder organisatorischen Einheit bezeichnet [werden], die ein Computersystem zur Erfüllung ihrer fachlichen Aufgaben einsetzen" und die Ergebnisse der Anwendungssoftware benutzen oder Daten liefern, die die Anwendungssoftware benötigt. Zum Kreis der Anwender gehören somit auch Personen, die indirekt durch die Anwendung der Software betroffen sein werden (vgl. [Alexander, 1999; Balzert, 2000a; Sommerville, 1996]). Häufig wird zudem angelehnt am englischen Begriff „End-User" die Bezeichnung End-Anwender bzw. End-Benutzer verwendet. Damit wird suggeriert, dass neben dem Anwender noch eine Gruppe der End-Anwender existiert, die am Ende der Wertschöpfung die eigentlichen Benutzer darstellen. Im Rahmen dieser Arbeit wird auf eine Unterscheidung zwischen End-Anwender und Anwender verzichtet. Stattdessen soll eine umfassende Definition des Anwenders aus dem VModell XT nach Broy und Rausch [2005] Anwendung finden:

> *„Der Anwender nutzt das System zur Erfüllung seiner Fachaufgaben nach der Auslieferung. Er leitet aus seiner Erfahrung mit dem Einsatz und Betrieb sowie der Pflege und Wartung von Systemen Anforderungen an das Gesamtsystem ab und bringt entsprechende Änderungsvorschläge ein."*

Somit bezeichnet der Anwender die Person, die die geplante Software nach ihrer Umsetzung und Inbetriebnahme nutzen wird. In einem Software-Projekt ist er in erster Linie für die Definition von Anforderungen und Änderungsvorschlägen verantwortlich. Er bringt dabei seine Meinung ein und ist Stellvertreter einer Anwender- bzw. Zielgruppe. Die Gruppe der Anwender ist allerdings nicht als homogene Masse zu verstehen. Die Anwender können aus einem definierbaren Personenkreis innerhalb eines Unternehmens stammen, als eine bestimmte Anwendergruppe eines Auftraggebers oder auch als anonyme Kundengruppe auf Konsummärkten verstanden werden.

Eine besondere Rolle nimmt der Anwender in der Open-Source-Software-Entwicklung ein. Im Fall eines Open-Source-Software-Projektes gehören die Anwender häufig auch zur Gruppe der Entwickler der Software. Die Organisation der Zusammenarbeit unterscheidet sich sehr stark von der Organisation anderer Arten von Software-Projekten, da die Entwicklung häufig räumlich und zeitlich verteilt über das Internet erfolgt und speziellen Entwicklungsparadigmen unterliegt. Kommunikationsmittel sind Email, Instant-Messaging, Issue-Tracker und Diskussionsforen. In wenigen Fällen wird auch das

Telefon zur Kommunikation verwendet. Der Erfolg dieses Entwicklungsparadigmas beruht zum großen Teil darauf, dass die Entwickler aus Eigenbedarf programmieren und die Software nach ihren Vorstellungen anpassen bzw. erweitern können. Nach dem Prinzip des „Benevolent Dictators" („wohlmeinenden Diktators") und „Cathedral and the Bazaar" („Kathedrale und der Basar") existieren klare Strukturen des Umgangs zwischen Entwicklern und Anwendern (vgl. [Raymond, 2001]). Dabei nimmt der Entwickler im Regelfall auch die die Rolle des Moderators und des Entscheiders ein.

2.4.3 Entscheider

Die Entscheider stellen die Wirtschaftlichkeit und Umsetzbarkeit der Anwendung der Software sicher und legen Ziel und Funktionsumfang der geplanten Software fest. Ihnen fällt die Entscheidungskompetenz zu, welche Anforderungen an eine Software letztendlich in die Spezifikation aufgenommen werden. Üblicherweise wird diese Rolle von Projektleiter, Projektmanager und Produktmanager eingenommen, die sich in Form eines Projekt-Lenkungsausschusses organisieren und gemeinsam Entscheidung treffen. Bei Projekten im kleineren Rahmen ist die Rolle des Entscheiders meist auf eine einzige Person beschränkt. Im V-Modell XT nehmen Projektleiter und Projektmanager die Rolle des Entscheiders ein, wobei der Projektleiter die operative Leitung und der Projektmanager die Verantwortung für die Wirtschaftlichkeit und Umsetzbarkeit besitzen:

> *„Der »Projektleiter übernimmt die operative Leitung des Projektes. Er plant, koordiniert, überwacht und steuert den »Projektablauf, das Projektteam und das Projekt als Ganzes. Er hat damit die Aufgabe, die Projektergebnisse der anderen Projektmitglieder zu beobachten und gegebenenfalls Nachbesserungen von den Produktverantwortlichen anzufordern."*

> *„Der »Projektmanager hat die Verantwortung gegenüber seinen jeweiligen Vorgesetzten und dem »Lenkungsausschuss, ein Projekt wirtschaftlich und technisch erfolgreich zu planen, durchzuführen und abzuschließen."*

Dem Projektlenkungsausschuss kommt nach dem VModell-XT die Entscheidungskompetenz bei Eskalationsstufen und Entscheidungspunkten in einem Projekt zu:

> *„Der »Lenkungsausschuss ist das oberste Entscheidungsgremium der Projektorganisation. In ihm sollten alle Projektbeteiligten (stakeholder) in geeigneter Weise vertreten sein. Normalerweise ist der »Projektmanager für die »Projektfortschrittsentscheidungen verantwortlich, weit reichende Entscheidungen wie z.B. über den Abbruch des Projektes müssen jedoch an den Lenkungsausschuss eskaliert werden."*

2.4.4 Moderator

Die Rolle des Moderators bildet die Schnittstelle zwischen Anwender und Entscheider. Für diese Rolle findet sich in der Literatur keine eindeutige Definition. Die Literatur integriert die Rolle des Moderators im Regelfall auf Seiten des Entwicklerteams als Teil der Rolle des Anforderungsanalysten („requirements analyst"), des Entwerfers („designers") oder des Entwicklers („developer"). Weitere synonyme Bezeichnungen sind Anwendungsspezialist, Anforderungsanalytiker, Usability Agent, Systemanalytiker, User Manager, Kundenmanager, Usability Engineer, Requirements Engineer bzw. Requirements Specifier. Grudin [1991a] führt die Rolle des Mediators zur Schaffung einer Brücke zwischen Entwicklern und Anwendern ein. Ein Mediator kann seiner Meinung nach jede beliebige Rolle außer

Entwickler und Anwender sein, die Einfluss auf die Kommunikation zwischen Anwender und Entwickler nehmen kann (vgl. [Grudin, 1991a, S. 64]):

> *„Although I have primarily considered users and developers, projects generally involve other parties. These parties include other groups within the development and user organization, as well as external consultants, subcontractors, value-added resellers, independent software vendors, third-party developers, product user organizations, trade unions, and standards organization. (…) The roles of such mediators are sometimes central, sometimes incidental; they include informing developers of users' needs and informing users of technological opportunities."*

Beim VModell-XT fällt die Aufgabe des Moderators in das Gebiet des Anforderungsanalytikers. Dabei wird unterschieden, ob dieser auf Seiten des Auftraggebers oder Auftragnehmers eines Software-Projektes agiert. Zu Beginn des Projektes, z.B. bei der Ausschreibung, ist der Moderator auf Auftraggeberseite vorzufinden. Mit der Beauftragung des Software-Projektes wird die Rolle des Moderators von der Seite der Entwickler besetzt. Auf Seiten des Auftraggebers wird gefordert:

> *„Der »Anforderungsanalytiker (AG) ist nach Erteilung des Projektauftrags für die Erstellung der »Produkte »Anforderungen (Lastenheft) und »Anforderungsbewertung zuständig. Bei Bedarf führt er zusätzlich eine »Marktsichtung für Fertigprodukte durch. Deren Ergebnisse werden im Rahmen der »Anforderungsbewertung evaluiert und entsprechend berücksichtigt, analog einer »Make-or-Buy-Entscheidung. Er hat die Qualität der Anwenderanforderungen sicherzustellen und die Voraussetzungen für die Verfolgbarkeit und die Veränderbarkeit der Anforderungen über alle Lebenszyklusabschnitte zu schaffen. Der Anforderungsanalytiker (AG) hat die Grundlagen der Fachgebiete ‚Requirements Engineering' und ‚Procurement Planning' bei der Aufgabendurchführung zu beachten."*

Die Rolle des Anforderungsanalytikers auf Seiten des Auftragnehmers umfasst:

> *„Der »Anforderungsanalytiker (AN) ist nach Erhalt der Anwenderanforderungen (Lastenheft) für die Erstellung des »Produktes »Gesamtsystemspezifikation (Pflichtenheft) zuständig. Für diese komplexe Aufgabe hat er fachspezifische Mitarbeiter einzubinden, um die Qualität der Anforderungen sicherzustellen und die Voraussetzungen für die Verfolgbarkeit aller Anforderungen über alle Lebenszyklusabschnitte zu schaffen. Der Anforderungsanalytiker (AN) hat die Grundlagen des Fachgebietes Requirements Engineering bei der Aufgabendurchführung zu beachten."*

Zusammengefasst ist die Rolle des Moderators folgendermaßen zu definieren: Im Aufgabenbereich des Moderators liegen die Erhebung von Anforderungen der Anwender und die Übersetzung dieser Anforderungen mithilfe von Software- bzw. Anforderungsspezifikationen (z.B. Lastenheft, Pflichtenheft, etc.) in die Sprache der Entscheider und Entwickler.

2.5 Anforderungsdokumente in Software-Projekten

Zentrales Ergebnis der Anforderungsanalyse ist die Spezifikation von Anforderungen an die geplante Software. Im IEEE Guide for Developing System Requirements Specifications [Tripp, 1998] bzw. IEEE

Standard Glossary of Software Engineering Terminology unter dem Kürzel IEEE Std 610.12-1990 wird folgende Definition einer Anforderung gegeben:

> *"(A) condition or capability needed by a user to solve a problem or achieve an objective.*
>
> *(B) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.*
>
> *(C) A documented representation of a condition or capability as in definition (A) or (B). (IEEE Std 610.12-1990)"*

Eine Anforderung ist somit die Beschreibung einer Funktionalität einer Software, die zur Erfüllung eines Ziels oder zur Lösung eines Problems eines Anwenders verhilft. Nach Balzert [2000a, S. 98] legen Anforderungen „*die qualitativen und quantitativen Eigenschaften eines Produkts aus der Sicht des Auftraggebers fest*". Ähnlich definiert Rupp [2001] eine Anforderung als „*eine Aussage über eine zu erfüllende Eigenschaft oder zu erbringende Leistung eines Produktes, Prozesses oder der am Prozess beteiligten Personen*". Sommerville [1996, S. 64f] unterscheidet Anforderungen anhand ihres Abstraktionsniveaus und ihres Zwecks und unterteilt diese in „requirements definition" und „requirements specification":

> *„(1) A requirements definition is a statement, in a natural language plus diagrams, of what services the system is expected to provide and the constraints under which it must operate. It is generated using customer-supplied information.*
>
> *(2) A requirements specification is a structured document which sets out the system services in detail. This document, which is sometimes called a functional specification, should be precise. It may serve as a contract between the system buyer and software developer."*

Darüber hinaus unterscheidet Sommerville zwischen „user requirements", „system requirements" und „domain requirements", mit denen Anforderungen auf Ebene der Anwender, des Systems und der Domäne verstanden werden. „User requirements" bzw. Anwenderanforderungen sind nach Sommerville [1996] in natürlicher Sprache formulierte Aussagen auf einem hohen Abstraktionslevel.

In der deutschen Literatur wird analog zu Sommervilles [1996] Verständnis von „requirements definition" und „requirements specification" zwischen dem Lastenheft und dem Pflichtenheft unterschieden. Das Lastenheft ist nach Balzert [2000a] eine Sammlung von Anforderungen, die aus Sicht des Kunden beschrieben wurden und ein grobe Umschreibung der geplanten Software enthalten. In der englischen Literatur wird dieser Detailgrad einer Anforderung als „r*equirement definition*" (vgl. [Sommerville, 1996, S. 64]) bezeichnet. Im V-Modell XT findet sich eine exakte Definition zum Lastenheft:

> *„Das Produkt Anforderungen (Lastenheft) enthält alle an das zu entwickelnde System verbindlich gestellten Anforderungen. Es ist Grundlage für »Ausschreibung und Vertragsgestaltung und damit wichtigste Vorgabe für die Angebotserstellung. Das Lastenheft ist Bestandteil des »Vertrags zwischen Auftraggeber und Auftragnehmer. Mit den Anforderungen werden die Rahmenbedingungen für die Entwicklung festgelegt, die dann vom*

> *Auftragnehmer in der »Gesamtsystemspezifikation (Pflichtenheft) detailliert ausgestaltet werden.“*

Aufbauend auf das Lastenheft wird das Pflichtenheft entwickelt, das die Spezifikation des Systems enthält und die Grundlage der weiteren Entwicklung bildet. Nach Balzert [2000a] stellt dieses Dokument eine ausführliche Beschreibung der geplanten Software dar, die auch technische Details und die geforderten Qualitätseigenschaften enthält. Nicht zuletzt bildet dieses Dokument häufig die vertragliche Vereinbarung zwischen Auftraggebern und Auftragnehmern eines Software-Projekts. In der englischen Literatur wird die Bezeichnung „*requirement specification*“ für diesen Detailierungsgrad von Anforderungen (vgl. [Sommerville, 1996, S. 64]) verwendet. Als Bezeichnung für das gesamte Anforderungsdokument wird der Name „software requirements document“ verwendet, welches zugleich „requirement definition“ und „requirement specification“ enthält. Eine exakte Definition zum Pflichtenheft stellt das VModell XT:

> *„Die »Gesamtsystemspezifikation (Pflichtenheft) ist das Pendant zu dem Auftraggeberprodukt Anforderungen (Lastenheft) auf Auftragnehmerseite. Sie wird vom Auftragnehmer in Zusammenarbeit mit dem Auftraggeber erstellt und stellt das zentrale Ausgangsdokument der Systemerstellung dar. (…)“*

Zudem wird zwischen funktionalen und nicht-funktionalen Anforderungen unterschieden. Die Grenze zwischen beiden Arten von Anforderungen lässt sich allerdings nicht immer klar ziehen und ist Thema zahlreicher Diskussionen (vgl. [Davis, 1990; Glinz, 2007]). Funktionale Anforderungen beschreiben grundsätzlich, was eine Software an Funktionalität bieten soll. Nicht-funktionale Anforderungen hingegen definieren, wie eine Funktionalität umgesetzt werden soll. Sie umfassen nach dem IEEE Recommended Practice for Software Requirements Specifications Aussagen u.a. zu Performanz, Schnittstellen, Akzeptanz, Dokumentation, Portabilität, Sicherheit, Qualität, Zuverlässigkeit, Ressourcen, etc.

Als Qualitätsanforderungen an eine hochwertige Anforderungsbeschreibung bzw. an hochwertige Anforderungsdokumente werden im IEEE Guide for Developing System Requirements Specifications neun Kriterien aufgeführt, die in Tabelle 2.2 aufgelistet werden. Dabei wird „SyRS“ als Abkürzung für „System Requirements Specification“ verwendet und beschreibt die Sammlung von Anforderungen an ein System bzw. eine Software. Dabei sollte jede Anforderung eindeutig beschrieben werden. Verbindungen zwischen Anforderungen sollten explizit markiert werden. Außerdem sollten in einem Anforderungsdokument der Detaillierungsgrad der Anforderungsbeschreibung definiert, die Anforderungen durchgängig konsistent beschrieben und alle identifizierten Anforderungen enthalten sein. Nicht zuletzt wird gefordert, den Fokus und die Grenzen des Anforderungsdokuments darzustellen und eine Anpassbarkeit und Versionisierung des Dokuments sicherzustellen.

Um als gut formulierte Anforderungen (“well-formed requirements”) bezeichnet werden zu können, fordert IEEE Guide for Developing System Requirements Specifications, dass eine Anforderung auf ihre Erfüllung validiert werden kann, der Definition einer Anforderung entspricht, messbare Eigenschaften der beschrieben Funktionalität enthält und die relevanten Rahmenbedingungen aufführt:

> *“A statement of system functionality (a capability) that can be validated, and that must be met or possessed by a system to solve a customer problem or to achieve a customer objective, and is qualified by measurable conditions and bounded by constraints.”*

Merkmal	Beschreibung
Einzigartigkeit („Unique set")	Jede Anforderung sollte nur ein einziges Mal definiert werden. („Each requirement should be stated only once.")
Normalisiert („Normalized")	Anforderungen sollten sich nicht überdecken. („Requirements should not overlap.")
Relationen („Linked set")	Verbindungen zwischen Anforderungen sollten explizit beschrieben werden. („Explicit relationships should be defined among individual requirements to show how the requirements are related to form a complete system.")
Vollständigkeit („Complete")	Die Spezifikation sollte alle Anforderungen des Kunden enthalten. („A SyRS should include all the requirements identified by the customer, as well as those needed for the definition of the system.")
Konsistenz („Consistent")	Inhalte der Spezifikation sollten in ihrem Detailgrad, in ihrer Beschreibungsform und in ihrer Darstellung konsistent sein. („SyRS content should be consistent and non-contradictory in the level of detail, style of requirement statements, and in the presentation of material.")
Kontext („Bounded")	Der Fokus und der Kontext der Anforderungen sollten definiert sein. („The boundaries, scope, and context for the set of requirements should be identified.")
Veränderbarkeit („Modifiable")	Die Spezifikation sollte nachträglich veränderbar sein. („The SyRS should be modifiable. Clarity and non-overlapping requirements contribute to this.")
Konfigurierbarkeit („Configurable")	Versionierungen der Spezifikation sollten nachvollziehbar sein. („Versions should be maintained across time and across instances of the SyRS.")
Granularität („Granular")	Der Abstraktionslevel der Spezifikation sollte definiert sein. („This should be the level of abstraction for the system being defined.")

Tabelle 2.2: Kriterien nach IEEE Guide for Developing System Requirements Specifications

Die Definition bzw. Spezifikation von Anforderungen durchläuft nach Sommerville [1996] einen mehrstufigen Entwicklungsprozess. Zu Beginn liegen die Anforderungen nach der Anforderungserhebung in unstrukturierter Form vor und müssen im weiteren Verlauf sortiert und strukturiert werden. Hierzu sind Standardformate bzw. Schemata für die Anforderungsspezifikationen hilfreich. Zur strukturierten Beschreibung von Anforderungen existieren mehrere unterschiedliche Optionen. Am häufigsten werden Anforderungen in natürlicher Sprache beschrieben. Dies bringt allerdings den Nachteil mit sich, dass natürliche Sprache zu Missverständnissen sowie zu einer ineffizienten und fehlerbehafteten Arbeitsweise führen kann (vgl. [Sommerville, 1996]). Als Alternativen empfiehlt Sommerville [1996] folgende Ansätze:

- Strukturierte natürliche Sprache („structured natural language“): Mithilfe einer Vorlage oder eines Schemas werden die Anforderungen strukturiert beschrieben. Das Volare-Template (vgl. [Robertson und Robertson, 1998; Robertson und Robertson, 2005]), UML Use Case Diagramme (vgl. [Booch et al., 1999]) und Entscheidungstabellen (vgl. [Moret, 1982]) sind bekannte Beispiele für diesen Ansatz.

- Entwurf-beschreibende Sprachen („design description languages“): Ähnlich wie eine Programmiersprache werden die Anforderungen mithilfe eines abstrakten Sprachmodells zur Beschreibung eines Systems bzw. einer Software definiert. Beispiele sind PDL („program description language“) und Pseudo-Code (vgl. [Balzert, 2000a]).

- Anforderungsspezifikationssprachen („requirements specification languages“): Zur Beschreibung von Anforderungen wurden zahlreiche Sprachkonstrukte entwickelt, wie z.B. PSL/PSA (vgl. [Teichrow und Hershey, 1977]) und RSL (vgl. [Alford, 1977]). Mit einem solchen Modell sollte die Möglichkeit für eine Werkzeugunterstützung der Anforderungsspezifikation geschaffen werden (vgl. [Bell et al., 1977]). Allerdings fand dieser Ansatz keine breite Verwendung.

- Grafische Notationen („graphical notations“): Mithilfe eines grafischen Vokabulars können Anforderungen grafisch skizziert werden. Beispiele sind u.a. SADT (vgl. [Ross, 1977; Schoman und Ross, 1977]), Datenflussdiagramme (vgl. [DeMarco, 1979]) und Objektdiagramme (vgl. [Booch, 1994]).

- Mathematische bzw. formale Notationen („mathematical specifications“): Formale Notationssprachen wie z.B. Petri-Netze (vgl. [Peterson, 1977]) oder endliche Automaten (vgl. [Gill, 1962]) stellen eine weitere Möglichkeit dar, Anforderungen basierend auf mathematischen Konzepten zu spezifizieren.

Bei der Spezifikation von Anforderungen ist darüber hinaus zu beachten, dass sich die Anforderungen während der Entwicklung ändern können und daher auch die Abhängigkeiten zwischen Anforderungen in einer Spezifikation Berücksichtigung finden sollten (vgl. [Sommerville, 1996]). Hierzu sollten nach Sommerville [1996] alle Anforderungen mit einer eindeutigen Zahl nummeriert und Beziehungen zwischen Anforderungen basierend auf dieser Nummerierung markiert werden. Außerdem sollte im Anforderungsdokument eine Matrix abgebildet werden, die alle Beziehungen bzw. Abhängigkeiten zwischen den Anforderungen aufzeigt.

Formal werden Änderungen als Anfrage zur Anforderungsänderung („Change Request“) formuliert und im Anforderungsdokument ergänzt. Ein Beispiel für den Ablauf eines solchen formalen Prozesses beschreibt Mohan et al. [2006]. Dabei wird bei Auftreten eines Problems eine Entscheidung getroffen, die eine Anfrage bezüglich einer Anforderungsänderung auslöst. Diese Anfrage führt zur Änderung des Anforderungsdokuments und damit auch zu einer neuen Version des Dokuments.

Von einer Anforderungsänderung abzugrenzen ist der Begriff des Fehlers („bugs“). Fehler sind Störungen in der Software und bedeuten, dass eine oder mehrere Anforderungen nicht erfüllt sind. Sie werden während der Tests entdeckt und als Fehlermeldung an die Entwickler kommuniziert. Die Fehlermeldungen werden üblicherweise in einer Liste („Bugtracker“) gesammelt und in der Reihenfolge ihrer Priorität abgearbeitet. Empfehlungen für gute Fehlermeldungen sind in Beiträgen von

Goldberg [2009], Spolsky [2004], Bettenburg et al. [2008] und Hooimeijer und Weimer [2007] enthalten. Fehlermeldungen sollten vor allem beschreiben, welcher Fehler aufgetreten ist und wie dieser entstand. Zusätzliche Daten wie Fehlercodes in Form von Stapeldateien („Stack Traces“) (vgl. [Pfleeger, 1998]) und Bildschirmfotos werden von Entwicklern als sehr hilfreich bewertet. Je besser ein Fehler reproduziert werden kann, desto einfacher gestaltet sich die Fehlersuche. Zudem sollte eine Fehlerbeschreibung so einfach wie möglich zu lesen und zu verstehen sein.

2.6 Formen der Anwenderbeteiligung in Software-Projekten

In der Literatur existiert keine eindeutige Definition des Terms Anwenderbeteiligung („user involvement“) (vgl. [Kujala, 2008]). Synonym zu Anwenderbeteiligung werden „Anwenderorientierung“ (vgl. [Lauenroth und Riechert, 2009]), „Partizipation“ (vgl. [Lohmann und Ziegler, 2008; Stevens und Draxler, 2006]), „Einbindung von Anwendern“ (vgl. [Lettl, 2004]) sowie vereinzelt „Anwenderfokus“ und „Anwenderzentrierung“ verwendet. Auch in der englischen Literatur existieren zu „user involvement“ unterschiedliche Termini (vgl. [Kujala et al., 2005]), u.a. „focus on users” (vgl. [Wilson et al., 1997]), „consulting end-users” (vgl. [Noyes et al., 1996]), „contacting with system users” (vgl. [Grudin, 1991a]) und „participation of users” (vgl. [Heinbokel et al., 1996]). Barki und Hartwick [1994] unterscheiden zwischen „user participation“ und „user involvement”. Mit „user participation“ können sich Anwender in ein Software-Projekt einbringen, während sich „user involvement“ auf die Mentalität der Anwender bezieht und die Wichtigkeit und persönliche Relevanz eines Software-Projektes für den Anwender bezeichnet.

Für diese Arbeit soll Anwenderbeteiligung grundsätzlich als jede Aktivität verstanden werden, die den Anwender aktiv in ein Software-Projekt einbindet und ihm Möglichkeiten zur Abgabe von Beiträgen bietet. Nach Kujala et al. [2005] beschreibt die Anwenderbeteiligung somit den direkten Kontakt zum Anwender mithilfe von methodischen Ansätzen:

> *„‘User involvement’ can be seen to be a general term describing direct contact with users and covering many approaches.“*

Nach Ortlieb und Holz auf der Heide [1993] sowie Damodaran [1996] können Anwender auf drei unterschiedliche Formen beteiligt werden:

- Passive Mitwirkung: Der Anwender stellt Informationen über seine Arbeitsumgebung zur Verfügung und kann seine Meinung zu vorgegeben Fragestellungen abgeben. Das Entwicklerteam berücksichtigt die Vorschläge nach eigener Einschätzung. Nach Damodaran [1996] kann diese Form der Beteiligung als konsultative bzw. bewertende Mitwirkung bezeichnet werden.
- Aktive Mitentscheidung: Der Anwender ist an Entscheidungen bezüglich Entwurf und Gestaltung beteiligt.
- Aktive Partizipation: Der Anwender kann aktiv mitentscheiden und ist gestaltend tätig.

Der Beitrag der Anwender kann somit entweder eine Meinung bzw. Information, eine Bewertung, eine Entscheidung oder ein Gestaltungsartefakt sein. Außerdem können Anwender indirekt beteiligt

werden wie z.B. bei Marktanalysen. Die Anwender wirken dabei indirekt durch ihr Verhalten oder die Benutzung von Software mit, ohne dass sie es selbst bemerken oder dazu befragt werden. Als Zeitpunkte der Anwenderbeteiligung werden in Software-Projekten primär die Anforderungsanalyse, der Test und die Einführungsphase gewählt (vgl. [El Emam et al., 1996]). In den letzten Jahrzehnten hat sich zudem die Einbindung der Anwender in der Entwurfs- und Implementierungsphase als erfolgskritisch erwiesen (vgl. [Kujala, 2003]).

Primäres Ziel der Anwenderbeteiligung ist die Sicherstellung der Gebrauchstauglichkeit (in Engl.: „Usability"). Der Begriff der Gebrauchstauglichkeit ist in der Norm DIN EN ISO 9241 (vgl. [DIN, 2004, S. 94]) definiert als

> *„das Ausmaß, in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen".*

Sie stellt somit das Qualitätsmaß an einer Software aus der Perspektive der Anwender dar. Mit Effektivität wird die Genauigkeit und Vollständigkeit bezeichnet, mit der ein bestimmtes Ziel erreicht werden kann. Die Effizienz beschreibt die Relation zwischen dem notwendigen Aufwand und der erforderlichen Genauigkeit und Vollständigkeit der Zielerreichung. Die Zufriedenheit umfasst die subjektive Wahrnehmung der Anwender. Sie beinhaltet die allgemeine Einstellung gegenüber der Systemnutzung und dem Grad der uneingeschränkten Systemnutzung.

In Abbildung 2.1 wird der Zusammenhang zwischen Gebrauchstauglichkeit und Nutzungskontext gemäß der DIN Norm illustriert. Die Gebrauchstauglichkeit hängt vom jeweiligen Nutzungskontext ab. Der Nutzungskontext umfasst den Benutzer bzw. Anwender selbst, dessen Arbeitsaufgabe und –mittel sowie dessen physische und soziale Umgebung. Mit der Nutzung des Arbeitssystems und des Produktes bzw. des Systems strebt der Anwender bestimmte Ziele an. Das Ergebnis der Nutzung drückt sich schließlich in der Gebrauchstauglichkeit aus.

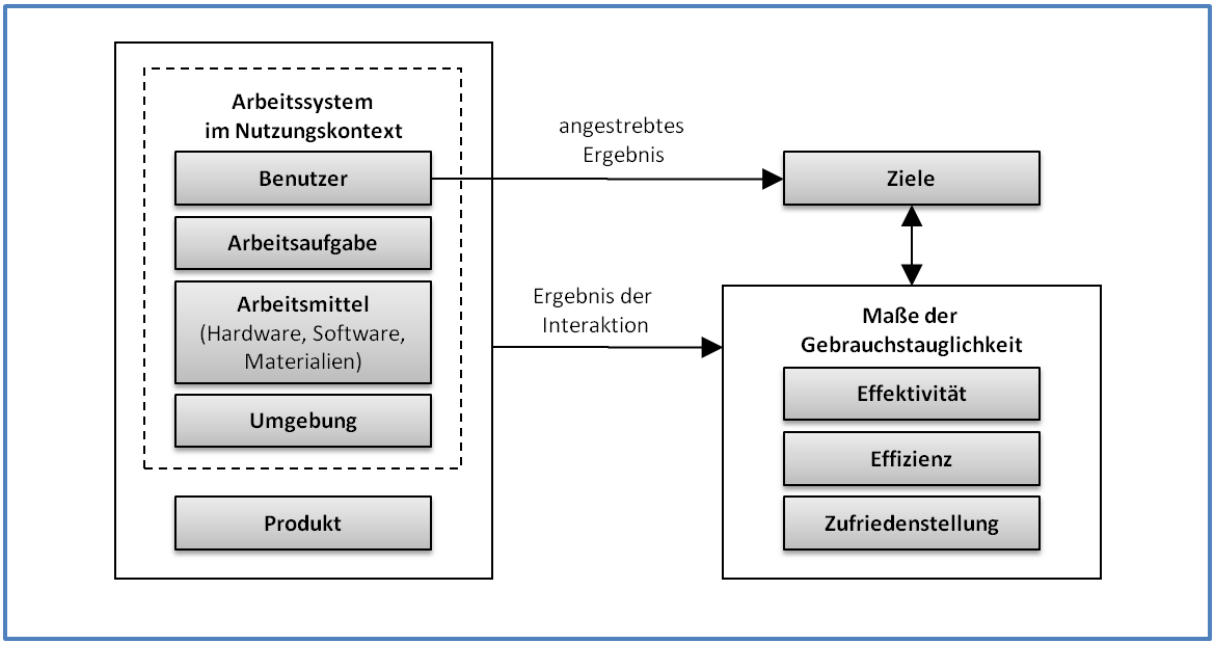

Abbildung 2.1: Nutzungskontext und Gebrauchstauglichkeit [DIN, 2004]

Für die Erreichung der Ziele der Anwenderbeteiligung, sind Methoden zur Anwenderbeteiligung anzuwenden. In Anlehnung an Menne [1984] beinhalten Methoden grundsätzlich eine Beschreibung des Vorgehens und der Voraussetzung ihrer Anwendung, um ein bestimmtes Ziel zu erreichen. Nach Greiffenberg [2003a, S. 31; 2003b] bildet die Anwendung von Methoden „den Ausgangspunkt ingenieurmäßigen Vorgehens“ in der Wirtschaftsinformatik. Sie sind gekennzeichnet durch eine Zielorientierung bzw. der Anforderung an ihre Ergebnisse, enthalten eine Anleitung für ihre Anwendung und beschreiben ein systematisches Vorgehen mit der Beschreibung von konkreten Aufgaben und Aufgabenträgern. Diese Arbeit übernimmt folgende Definition von Methoden, Werkzeugen, Verfahren und Prinzipien nach Hesse et al. [1992]:

- *„Methoden sind planmäßig angewandte, begründete Vorgehensweisen zur Erreichung von festgelegten Zielen (i.a. im Rahmen festgelegter Prinzipien). Methoden können fachspezifisch sein“*
- *„Werkzeuge dienen der automatisierten Unterstützung von Methoden und Verfahren.“*
- *„Verfahren sind ausführbare Vorschriften oder Anweisungen zum gezielten Einsatz von Methoden. Eine Methode kann durch mehrere (alternative oder sich gegenseitig ergänzende) Verfahren unterstützt werden.“*
- *„Prinzipien sind Grundsätze, die man seinem Handeln zugrunde legt. Solche Grundsätze sind i.a. nicht nur für ein bestimmtes Teilgebiet, sondern für das gesamte Fachgebiet oder zumindest wesentliche Teile davon, womöglich auch über das Fachgebiet hinaus im wissenschaftlich-technischen Bereich gültig.“*

Da der Begriff der Methode zur Anwenderbeteiligung auf unterschiedlichen Abstraktionsebenen verwendet und in der Literatur häufig mit Partizipationsmodellen gleichgesetzt werden, wird in dieser Arbeit zudem zwischen Methoden zur Anwenderbeteiligung und Partizipationsmodell unterschieden:

- Eine Methode zur Anwenderbeteiligung beschreibt eine Verfahrensweise zur Interaktion zwischen Anwender, Entwickler, Moderator und Entscheider (vgl. [Carrizo et al., 2008; Goguen und Linde, 1993; Keil und Carmel, 1995; Sarodnick und Brau, 2006]). Sie dienen der Erfassung von Informationen von Anwendern und der Einbindung der Anwender in Gestaltungs- und Entscheidungsprozesse von Software-Projekten.
- Ein Partizipationsmodell ist ein Prinzip und umfasst eine komplexe nicht deterministische Vorgehensweise, die auch eine Kombination ausgewählter Methoden für ein effektives Zusammenspiel der Methoden in einem Projekt empfehlen. Beispiele für Partizipationsmodelle sind CARD „Collaborative Analysis of Requirements and Design“ (CARD, vgl. [Muller et al., 1995]), „Plastic Interface for Collaborative Technology Initiatives through Video Exploration” (PICTIVE, vgl. [Muller, 1991; Muller, 1993]), „Cooperation with Users in Task Analysis for Requirements and Design” (CUSTARD, vgl. [O’Neill et al., 1995]), „Cooperative Experimental Systems Development” (CESD, vgl. [Grønbæk et al., 1997]), „Effective Technical & Human Implementation of Computer-based Systems” (ETHICS, vgl. [Mumford, 1995]), „Soft Systems Methodology“ (SSM, vgl. [Checkland, 1981]), „Joint Application Design/ Development” (JAD, vgl. [Wood und Silver, 1995]) und „MUST” (vgl. [Kensing et al., 1998]).

2.7 Erfolgsfaktoren der Anwenderbeteiligung

Eine Fülle an Erfolgsfaktoren bzw. Best-Practices für eine bestmögliche Anwenderbeteiligung sind im Rahmen der Arbeiten zu den Paradigmen User-Centered-Design (UCD), Ethnografischen Methoden und Participatory Design (PD) entwickelt worden. Alle drei Paradigmen stellen unterschiedliche Bewegungen aus Forschung und Industrie dar, die sich auch gegenseitig beeinflussen und sich in einigen Punkten überschneiden. Allerdings unterscheiden sie sich stark in ihren Ausprägungen und ihren Schwerpunkten. Während sich UCD auf den Nutzungskontext („context of use") fokussiert, misst PD der Entscheidungs- und Handlungskompetenz der Anwender große Bedeutung bei. Ethnografische Methoden hingegen setzen sich insbesondere mit sozialen Aspekten der menschlichen Zusammenarbeit auseinander. Da alle drei Ansätze auch den Stand der Technik zur Anwenderbeteiligung in Software-Projekten darstellen, werden sie im Folgenden näher erläutert. Darauf aufbauend wird anschließend eine Zusammenfassung von Erfolgsfaktoren der Anwenderbeteiligung gegeben.

2.7.1 Erfolgsfaktoren des User-Centered Design (UCD)

Erstmalig aufgeführt wurde UCD von Gould und Lewis [1985], Norman [1986] und Knittle et al. [1986] und durchlief im Laufe der Jahrzehnte zahlreiche Wandlungen und Erweiterungen. In der Literatur werden zudem häufig synonym die Bezeichnungen User-Centered Systems Design (UCSD) und Human-Centered Design (HCD) verwendet. Mit UCD werden Qualitätssteigerung durch bessere Anwenderanforderungen, Kostensenkungen durch Fokussierung auf die relevante Funktionalität, Akzeptanzsteigerung durch frühe Beteiligung der Anwender, Effektivitätssteigerung der späteren Nutzung durch bessere Anpassung an die Arbeitsabläufe und eine erhöhte aktive Anwenderbeteiligung durch aktive Partizipation angestrebt (vgl. [Damodaran, 1996]). Ferner sollen auch die Kosten für Schulungen und Support reduziert werden können (vgl. [Dray und Karat, 1994; Kujala, 2003]).

Nach DIN EN ISO 13407 [1999] zu „Human-centered design processes for interactive systems" ist UCD durch vier Prinzipien charakterisiert, die so genannten „*four principles:*

1. *The active involvement of users and a clear understanding of user and task requirements;*
2. *An appropriate allocation of function between users and technology;*
3. *Iteration of design solutions;*
4. *Multi-disciplinary design."*

Diese Norm setzt somit eine aktive Beteiligung der späteren Anwender voraus, die über eine bloße Befragung oder Beobachtung zu Projektbeginn hinausgeht. Die Effektivität der Anwenderbeteiligung nimmt nach der Norm mit wachsender Interaktion zwischen Anwendern und Entwicklern zu. Sie sollte sich somit in unterschiedlicher Form bei den verschiedenen Projektaktivitäten über den gesamten Entwicklungsprozess erstrecken.

Die Beteiligung muss der Norm nach fest im Vorgehensmodell der Software-Entwicklung verankert sein. Sich wiederholende Entwurf-Test-Zyklen, in denen jeweils ein Prototyp bzw. eine Zwischenstufe des zu entwickelnden Programms erstellt und von den Anwendern überprüft wird, sind die Kernelemente eines solchen Vorgehensmodells. Die Auswahl der beteiligten Anwender ist so zu treffen, dass

ihre Kenntnisse möglichst alle relevanten Aspekte des Nutzungskontexts abdecken. Die Norm betont, dass eine Vielzahl von Kenntnissen und Fähigkeiten im Rahmen eines benutzerorientierten Gestaltungsprozesses erforderlich sind, also "multidisziplinäre Gruppen" zu beteiligen sind. Dabei sind insbesondere sowohl solche Personen zu berücksichtigen, die über das relevante Anwendungswissen verfügen, als auch solche, die auf Seiten der Entwickler für die technische Entwicklung zuständig sind. Zudem sollte nur eine repräsentative Auswahl von Mitarbeitern beteiligt werden. Die Arbeitsfähigkeit bei Gesprächsrunden oder Workshops ist ein wichtiges Kriterium. Kleine und dynamische Gruppen sind gemäß Norm vorzuziehen.

Aus Sicht mehrerer UCD-Forscher (vgl. [Gulliksen et al., 2003; Iivari und Iivari, 2006]). ist die Definition der ISO 13407 nicht präzise und eindeutig genug. Als Weiterentwicklung schlagen Gulliksen et al. [2003, S. 5ff] in Anlehnung an ISO 13407 und früheren Arbeiten zu UCD (vgl. [Gould et al., 1997]), zu Participatory Design (vgl. [Greenbaum und Kyng, 1991]), zu Contextual Design [Holtzblatt und Beyer, 1999], Goal-directed Design (vgl. [Cooper, 1999]) und Usability Engineering (vgl. [Nielsen, 1993]) zwölf grundlegende Prinzipien vor, die so genannten *„key principles*

1. *User focus – the goals of the activity, the work domain or context of use, the users' goals, tasks, and needs should early guide the development.*
2. *Active user involvement – representative users should actively participate, early and continuously throughout the entire development process and throughout the system lifecycle.*
3. *Evolutionary system development – the systems development should be both iterative and incremental.*
4. *Simple design representations – the design must be represented in such ways that it can be easily understood by users and all stakeholders.*
5. *Prototyping – early and continuously, prototypes should be used to visualize and evaluate ideas and design solutions in cooperation with end users.*
6. *Evaluate use in context – baselined usability goals and design criteria should control the development.*
7. *Explicit and conscious user interaction/interface design activities.*
8. *A professional attitude – the development process should be performed by effective multidisciplinary teams.*
9. *Usability champion - usability experts should be involved early and continuously throughout the development lifecycle.*
10. *Holistic design – all aspects that influence the future use situation should be developed in parallel.*
11. *Process customization – the UCSD process must be specific, adapted and/or implemented locally in each organization.*
12. *A user-centered attitude should always be established."*

Gulliksen et al. [2003] fordern somit von einer UCD-orientierten Entwicklung, dass früh im Projekt der Fokus auf die Bedürfnisse der Anwender gelegt und diese projektbegleitend aktiv berücksichtigt werden. Die Anwender sind früh und kontinuierlich in den gesamten Entwicklungsprozess einzubeziehen. Hierfür soll der Prozessablauf iterativ und inkrementell erfolgen. Durch eine leicht verständliche Darstellung des Entwurfs und den frühen Einsatz von Prototypen soll der aktuelle Stand der Entwicklungen in Zusammenarbeit mit Anwendern evaluiert und dabei auch der spätere Nutzungskontext berücksichtigt werden. Dem Anspruch der Gebrauchstauglichkeit („Usability“) soll durch ein multidisziplinäres und professionelles Team, das den Ansatz des UCD verinnerlicht hat und neben Anwendern und Entwicklern auch einen ausgewiesenen Usability Experten enthält, Rechnung getragen werden. Außerdem sollen in einem ganzheitlichen Ansatz auch die nicht-technischen Aspekte der Arbeitsumgebung der Anwender beachtet und an die neue Situation mit der geplanten Software während der Entwicklung angepasst werden. Ein UCD Prozess muss dabei spezifisch an die Rahmenbedingungen der einzelnen beteiligten Organisationen angepasst sein.

Einen anderen Blickwinkel nehmen Iivari und Iivari [2006] ein. UCD besteht aus ihrer Sicht aus vier Dimensionen, die den Grad der Anwenderbeteiligung („User-Centredness“) eines Software-Projektes definieren. Ihrer Auffassung nach würde UCD keine eigene Entwicklungsmethode darstellen sondern als vier-dimensionales Konzept in bestehende Entwicklungsmethoden integriert werden. Als Dimensionen des UCD sehen sie den Grad des Fokus auf die Anwender, des Fokus auf die Arbeitsumgebung, der Entscheidungskompetenz der Anwender und der Anpassbarkeit der Software. Mit dieser Sichtweise lassen sie es den Entwicklern offen, bis zu welchem Grad ihre Entwicklungsmethoden UCD-konform sind und erlauben im Gegensatz zu Gulliksen et al. [2003] auch Abweichungen. Dieses Verständnis von UCD spiegelt sich auch in früheren Definitionen, wie z.B. von Norman [1986] oder Karat [1997] wieder, die beide zwar ein gutes Verständnis über die Anforderungen der Anwender wünschen, aber den hierfür notwendigen Grad der Anwenderbeteiligung und Entscheidungskompetenz nicht vorschreiben.

Diese Unklarheit findet sich auch in der Praxis, wie es u.a. in einer Umfrage von Mao et al. [2005] zu UCD unter 103 UCD Praktikern festgestellt wurde. Diese ergab, dass UCD in Organisationen sehr unterschiedlich aufgefasst und umgesetzt wird. Dem Idealfall, wie es Gulliksen et al. [2003] vorschlägt, wurde in keinem der Fälle entsprochen. Zu Fragen zur Angemessenheit der Umsetzungstiefe von UCD wurden drei signifikante Antworten gegeben:

- Die Kosten zu UCD nehmen mehr als 10% des Budgets ihrer Software-Projekte ein. Dabei werden günstige und einfache Mittel (u.a. Usability Evaluation, Experten-Interview) gegenüber den aufwändigeren Methoden bevorzugt, obwohl den aufwändigeren Methoden wie z.B. Feldstudien ein sehr hoher Nutzen beigemessen wird. Das Kosten-Nutzen-Verhältnis der angewendeten Methoden ist ausschlaggebend.

- UCD konnte der einhelligen Meinung der Umfrageteilnehmer nach merklich zur Erhöhung der Nützlichkeit und Gebrauchstauglichkeit der Produkte und gleichzeitig auch zur Senkung von Entwicklungskosten beitragen.

- Allerdings werden die Effekte nur in wenigen Fällen tatsächlich gemessen, obwohl entsprechende Indikatoren und Konzepte existieren (vgl. [Vredenburg et al., 2001]). Als häufigster Indikator wurde die Kundenzufriedenheit gemessen. Auf den nachfolgenden Plätzen wurden

die verbesserte Gebrauchstauglichkeit, der Effekt auf den Umsatz und eine Verringerung der Menge an Helpdesk-Anfragen genannt. Die Mehrheit gab jedoch an, keine Indikatoren zu verwenden.

Zusammengefasst ist festzuhalten, dass UCD ein Paradigma der Anwenderbeteiligung der Praxis darstellt, das sich mit dem Prozess der Anwendereinbindung beschäftigt und eine ganzheitliche Integration des Anwenders anstrebt. Die unterschiedlichen Definitionen besitzen derzeit einen Empfehlungscharakter. In der Praxis ist die Umsetzungstiefe von UCD abhängig von der Einschätzung der Projektmitarbeiter und des Kosten-Nutzen-Verhältnisses.

Als Methoden des UCD werden häufig „Prototyping", „Fokusgruppen", „Contextual Inquiry" (vgl. [Holtzblatt und Jones, 1993]), „Interviews" , „Feldstudien" (vgl. [Bly, 1997; Wixon und Ramey, 1996; Wixon et al., 2002]), „Szenarios/ Anwendungsfälle" (vgl. [Alexander und Maiden, 2004; Carroll und Rosson, 1992]), „(Usability) Workshops" (vgl. [Nielsen, 1994a; Nielsen und Mack, 1994; Wixon et al., 1994]) und „Aufgabenanalyse" (vgl. [Diaper, 2001; Hackos und Redish, 1998]) genannt.

2.7.2 Erfolgsfaktoren der Ethnografischen Methoden

Die Ethnografie ist nach Meier [2001, S. 46] *„eine allgemeine Bezeichnung für eine meist monografische Beschreibung eines Volkes, einer Lebensweise oder einer Kultur"*. Nach Fetterman [1999] stellt sie die Kunst und Wissenschaft zur Beschreibung einer Gruppe oder einer Kultur dar. Während sich in der Anthropologie erste ethnografische Arbeiten (z.B. [Malinowski, 1922]) mit der Erforschung exotischer Inseln und Kulturen beschäftigten, wendeten Forscher der Sozialwissenschaften ethnografische Methoden bei der Untersuchung von sozialen Brennpunkten (z.B. [Jahoda et al., 1933; Jahoda et al., 1975]) an. Der Untersuchung liegt zentral eine aktive Teilnahme im Untersuchungsfeld zugrunde, bei der der Frage nach dem „Warum etwas passiert" nachgegangen wird [Fetterman, 1999; Lamnek, 2005].

Zentrale Charakteristika ethnografischer Feldforschung sind nach Meier [2001, S. 47f] eine explorative und langfristig-angelegte Vorgehensweise im unmittelbaren Umfeld des Untersuchungsgegenstands. Dabei können bzw. müssen Fragestellungen und Vorgehensweise an die im Feld gewonnen Erkenntnisse angemessen angepasst werden. Die Vorgehensweise konzentriert sich im Detail auf einzelne Fälle bzw. Felder einer bestimmten Gruppe von Menschen. Sie umfasst sowohl systematische Befragungen und Beobachtungen als auch unsystematische teilnehmende Beobachtungen, bei denen eine Gruppe von Menschen in ihrem Alltagsleben über einen langen Zeitabschnitt begleitet und beobachtet werden. Im Fokus der Untersuchungen steht das *„Verstehen(s) der Bedeutungen von Handlungen"* (vgl. [Meier, 2001, S. 47f]) und das *„soziale Handeln als sinnhaftes Tun"* (vgl. Meier [Meier, 2001, S. 47f]). Das Prüfen von Hypothesen ist nicht Gegenstand einer ethnografischen Untersuchung.

Die ersten ethnografischen Studien zu IUK-Technologien und Informationssystemen wurden in den 70er Jahren durchgeführt (vgl. [Meier, 2001]). Der Einsatz von ethnografischen Methoden wurde aufgrund der Erfolgslosigkeit früherer Software-Projekte, des steigenden Anspruchs an die Gebrauchstauglichkeit von Software und des zunehmenden Bedarfs an Systemen zur Unterstützung von Gruppen bzw. Gruppenarbeit, sog. CSCW Systemen in Erwägung gezogen. Es hat sich in den Untersuchungen gezeigt, dass für die Entwicklung von Systemen für die Gruppenarbeit ein tiefes Verständnis über die Art und Weise der Bewältigung von Arbeitsaufgaben im natürlichen Arbeitskontext notwen-

dig ist (vgl. [Meier, 2001]). Aufbauend auf dieses Wissen sollen Systeme entwickelt werden können, die sich in die Arbeitsprozesse einfügen, effektiver sind und auf höhere Akzeptanz der Anwender stoßen (vgl. [Blomberg et al., 1993]).

Ethnografischen Methoden (EM) wären in ihrer klassischen Form allerdings sehr zeitaufwändig und kostenintensiv. Sie können aufgrund des Kosten- und Zeitdrucks von Software-Projekten nur in sehr seltenen Fällen Einsatz finden. Daher werden nach Harper [2000] in Software-Projekten in der Regel leicht-gewichtige und fokussierte Ansätze der ethnografischen Methoden, wie z.B. „rapid ethnography“ (vgl. [Millen, 2000]) oder „quick & dirty ethnography“ (vgl. [Hughes et al., 1994]), verwendet. Die Beobachtungen belaufen sich auf wenige Wochen und auf einen eingegrenzten Fokus des Arbeitsumfelds.

Die Ergebnisse einer ethnografischen Studie münden in eine analytische Beschreibung der Beobachtungen und Befragungen in Form eines Textdokuments und können in einem Software-Projekt auf unterschiedliche Weise eingesetzt werden. Die Entwickler können aus der Beschreibung Rückschlüsse für die Anforderungen an der Software ziehen. Zudem können sie den Ethnografen als Advokaten der Anwender konsultieren. Außerdem wird durch eine ethnografische Studie erwartet, dass in dem untersuchten Feld „*eine Veränderungsdynamik in Gang kommt*“ (vgl. [Meier, 2001, S. 50]) und somit die Bereitschaft und Kompetenz der Anwender zur Beteiligung im Software-Projekt erhöht werden. Der Nutzen von ethnografischen Methoden lässt sich allerdings schwer quantifizieren (vgl. [Lamnek, 2005]).

Für den Ethnografen stellen sich bei der Durchführung einer ethnografischen Studie in einem Software-Projekt mehrere Herausforderungen. Er hat sich um eine innere Distanz zu seinem Wissen und seinen Vorstellungen zu bemühen, um auch implizite Praktiken und Wissen zu erkennen. Er nimmt die Rolle des professionellen Praktikanten (vgl. [Van Maanen und Kolb, 1986]) bzw. professionellen unerfahrenen Anwenders (vgl. [Nardi, 1997]), um mit methodischer Kompetenz und ohne Beeinflussung vorzugehen. Für die effektive und effiziente Durchführung einer ethnografischen Studie stellt sich zudem die Schwierigkeit bei der Bestimmung des Fokus, damit der Arbeitskontext umfassend mit angemessenem Zeit- und Kostenaufwand untersucht werden kann. In der Kommunikation mit dem Entwicklerteam ist zu berücksichtigen, dass sich die Arbeitsphilosophie des Ethnografen von der der anderen unterscheiden kann. Zwei Effekte können eine Zusammenarbeit erschweren:

1. Ethnografen empfehlen im Allgemeinen ein schrittweises Verbessern der Arbeitsumgebung statt einer radikalen Umstrukturierung von Prozessen. Dies kann zu einem Konflikt mit Entwicklern und Entscheidern führen, die mithilfe einer Software die bestmögliche Unterstützung der Arbeitsprozesse anstreben und hierfür auch radikale Umstrukturierungen in Kauf nehmen. Zudem zeigte sich auch, dass sich radikale Anpassungen nicht immer negativ auswirken.

2. Entwickler erwarten von den Ergebnissen der Ethnografen konkrete eindeutige Anhaltspunkte auf Verhaltens- und Arbeitsregeln für die Anwenderanforderungen an die Software, obwohl ethnografische Studien dies nicht liefern können. Daher sollten die Studien bereits früh in einem Projekt erfolgen und „*von vorherein auf die alltäglichen Praktiken im Arbeitsfeld und die Beantwortung konkreter Design-Fragen*“ (vgl. [Meier, 2001, S. 51]) fokussieren, um eine

Basis für grundlegende Annahmen über den Kontext der Anwender zu schaffen (vgl. [Harper und Simpson, 1998]).

Als Methoden werden in einer ethnografischen Studie „Befragungen“, „Beobachtungen“, „Protokoll- bzw. Dokumentanalyse“, „Aufgabenanalyse“ bzw. „Raum- und Werkzeugnutzungsanalyse“ und „Culture Probes“ bzw. „Tagebuchstudien“ eingesetzt (vgl. [Gaver, 1999]).

2.7.3 Erfolgsfaktoren des Participatory Design

Das Themenfeld des Participatory Design (PD) wurde in den frühen 70er Jahren erschlossen und entwickelte sich vom zunächst skandinavischen Forschungsprogramm zu einem international anerkannten Paradigma der Anwenderbeteiligung in Software-Projekten. PD entstand als Gegenreaktion auf die zunehmende Einführung von Computern an Arbeitsplätzen und den damit verbundenen Ängsten vor Arbeitslosigkeit, Überforderung, Kompetenzverlust, Kontrolle, etc. PD Forscher befürchteten, dass Computer zur Kontrolle und Entmachtung der Arbeiterschaft ausgenutzt und nicht zur Verbesserung der Arbeitssituation der Arbeiter (vgl. [Ehn und Sandberg, 1979; Kyng und Mathiassen, 1982]) verwendet werden.

Die Ursprünge von PD gehen auf erste Projekte im skandinavischen Raum, z.B. NJMF (vgl. [Nygaard, 1979]), DEMOS (vgl. [Ehn und Sandberg, 1979]), DUE (vgl. [Kyng und Mathiassen, 1982]), UTOPIA (vgl. [Bødker et al., 1987]) zurück, als die Politik und Gesetzgebung mehr Rechte für Arbeiter zur Einflussnahme auf Entscheidungsprozesse in Unternehmen schuf. Im Auftrag der Gewerkschaften wurde in mehreren Forschungsprojekten untersucht, wie Arbeiter effektiv Einfluss auf die Entwicklung von Computersystemen für ihren Arbeitsplatz nehmen können (vgl. [Kensing, 1983; Nygaard und Bergo, 1975]). Aufbauend auf den Vorarbeiten von Kensing [1983] untersuchten Clement und Van den Besselar [1993] zehn PD Projekte aus den 70ern und 80ern und erarbeiteten in ihren Analysen fünf Erfolgsfaktoren für PD-orientierte Software-Projekte:

- Anwender erhalten Zugang zu allen relevanten Informationen.
- Es wird eine unabhängige Position zu den Problemstellungen eingenommen.
- Anwender können an der Entscheidungsfindung teilnehmen.
- Es sind geeignete partizipative Entwicklungsmethoden verfügbar.
- Es wird Raum für alternative technische und organisatorische Maßnahmen eingeräumt.

Diese Anforderungen sollen Freiräume für Anwender und Entwickler für eine gute Zusammenarbeit und ein gutes Ergebnis schaffen. Partizipation kann dabei nach Gärtner und Wagner [1996] auf drei Ebenen angewendet werden:

- Arena A: Auf der individuellen Projektarena werden Software und neue Organisationsformen gestaltet.
- Arena B: Die Unternehmensarena umfasst die Einhaltung organisatorischer Vereinbarungen und die Gestaltung der organisatorischen Rahmenbedingungen.
- Arena C: Auf der nationalen Arena werden allgemeine rechtliche und politische Rahmenbedingungen geschaffen und Normen für arbeitsrelevante Aspekte gesetzt.

Die früheren PD Bemühungen der 70er und 80er Jahre bezogen sich auf alle drei Arenen, wobei aus Ergebnissen auf Arena A Rückschlüsse auf Arena B und C gezogen wurden. Ziel dieser Bemühungen war allerdings nicht die Verbesserung der Nützlichkeit und Gebrauchstauglichkeit von Software sondern die Demokratisierung des Entwurfs und der Umsetzung von Software und Prozessen am Arbeitsplatz. Das ausschließliche Kriterium von PD lag zum damaligen Zeitpunkt auf der Zufriedenheit des Anwenders mit dem Entwicklungsprozess. Die Güte der Software oder die Effizienz des Arbeitsprozesses wurden nicht untersucht; im Grunde wurden die Bedürfnisse der Arbeiter bzw. Anwender und der Gewerkschaften primär untersucht und Themen des Managements vermieden (vgl. [Bødker, 1996; Ehn, 1989; Ehn, 1993; Kensing und Blomberg, 1998]).

Dieser rein politische Ansatz geriet in den 80ern in die Kritik und wurde im Laufe folgender Forschungsarbeiten um weitere sozio-technische Aspekte erweitert. Insbesondere PD Forscher aus nicht-skandinavischen Ländern wie Nord-Amerika und Europa öffneten das Spektrum des PD um Aspekte der Wirtschaftlichkeit, da in diesen Ländern andere Unternehmens- und Arbeitskulturen existieren und die Rationalität des PD im Sinne eines Kosten-Nutzen-Vergleichs für Unternehmer Vorrang hatte (vgl. [Blomberg et al., 1997; Kensing und Blomberg, 1998; Muller et al., 1997]). PD erfuhr seinen internationalen Durchbruch, als Qualitätssteigerungen und Kostenreduktionen in unternehmerischen Prozessen mithilfe von PD nachgewiesen werden konnten (vgl. [Blomberg et al., 1993; Gärtner, 1998; Holtzblatt und Jones, 1993; Kensing und Blomberg, 1998; Muller, 1993; Muller et al., 1995]). Dies führte allerdings auch zu einer Konzentration der PD Forschung auf Arena A, der individuellen Projektarena und der Vernachlässigung der anderen beiden Arenen (vgl. [Beck, 1996; Bjerknes und Bratteteig, 1995; Gärtner, 1998; Greenbaum, 1995; Kensing und Blomberg, 1998; Kensing et al., 1998; Kujala, 2003; Kujala, 2008; Suchman, 1995]).

In diesem Zuge erfuhr das Verständnis von PD einen Wechsel von der skandinavisch geprägten politischen Motivation zu einer sozio-technischen Ausprägung, die sich auch stark von den Entwicklungen aus dem UCD beeinflussen ließ. Während in früheren Publikationen zu PD ausschließlich der Prozess der partizipativen Entwicklung diskutiert wurde, wird in späteren Publikationen zunehmend auch die Ergebnisqualität der partizipativen Entwicklung in Form von Software oder Prozessen besprochen (vgl. [Ehn, 1993; Greenbaum, 1995; Kensing und Blomberg, 1998]). Die demokratisch-motivierten Ziele wurden somit in PD schließlich als gleichwertig mit Produkt- und Entwicklungszielen betrachtet. Die Art der Anwenderbeteiligung in PD rangiert somit vom Extrem des Anwenders als reine Informationsquelle bis zum Extrem des Anwenders als Mitentscheider und -gestalter (vgl. [Kensing und Blomberg, 1998; Kujala, 2008]).

Zum aktuellen Zeitpunkt verfolgt PD (vgl. [Damodaran, 1996; Kujala, 2008; Muller et al., 1997] sowohl demokratisch-motivierte als auch organisatorisch-motivierte und entwicklungs-orientierte Ziele:

- Mit der demokratisch-motivierten Zielsetzung sollen durch PD die Arbeitnehmer bzw. Anwender an der Entscheidungsfindung teilnehmen und ihre Arbeitssituation beeinflussen können. Außerdem sollen die Kompetenz und Fähigkeiten der Arbeitnehmer gesteigert werden.

- Aus organisatorisch-motivierter Sichtweise sollen durch PD die Akzeptanz des Systems von den Anwendern geschaffen und in diesem Zuge auch die Produktivität der Anwender beim Einlernen und Anwenden erhöht werden.

- In Bezug auf Software-Projekte sollen durch den Einsatz von PD der Arbeitskontext der Anwender verstanden, genauere Anwenderanforderungen spezifiziert, die Qualität des Systems und die Effizienz der Entwicklung verbessert sowie die Anwender- und Kundenzufriedenheit erhöht werden.

Der Einsatz von Prototypen erfreut sich bei PD Praktikern und Forschern großer Beliebtheit und wird als Medium für die Evaluation von Gestaltungsmöglichkeiten und zur Gewinnung von Verbesserungsmöglichkeiten oder radikal neuen Ideen häufig eingesetzt. Mit der Verwendung von Prototypen in PD ändert sich nach Carey und Mason [1983, S. 179] allerdings auch das Machtverhältnis im Projekt zu Gunsten der Entwickler:

„The prototype system reflects the developer's interpretation of the user's needs."

Der Anwender erfährt durch das Prototyping den Nachteil, dass ein Prototyp aufgrund seiner fehlenden Programmier- und Anwendungskenntnissen im Regelfall nur vom Entwickler erstellt werden kann. Während der Entwickler seine Ideen mithilfe von Prototypen ausdrücken kann, kann der Anwender seine Ideen an Prototypen nur im Gespräch bzw. in Verhandlung mit dem Entwickler einbringen und ist damit auch von der Kooperations- und Interpretationsfähigkeit des Entwicklers abhängig. Als Gegenmaßnahmen bieten sich Schulungen der Anwender an, wobei allerdings dadurch die neutrale entwicklungsunabhängige Perspektive der Anwender beeinträchtigt werden kann. Einen anderen Ansatz verfolgt der Papier-Prototyp („Paper-Prototyping"), bei der Anwender ohne besondere Fähigkeiten eigene Prototypen mit geringem Aufwand erstellen können. Insgesamt wurde in den 80er Jahren die Kritik laut, dass die PD Forschung sich zu stark mit dem Einsatz von Prototypen beschäftigt und andere Möglichkeiten außen vor bleiben. In diesem Zusammenhang wurden in den 90er Jahren Methoden zur partizipativen bzw. kooperativen Analyse („participative analysis") unter der Bezeichnung „Contextual Inquiry" entwickelt, die im Vorfeld des Prototyping zur Analyse des Arbeitsumfelds und der Bedürfnisse der Anwender eingesetzt werden. Hierbei bediente sich die PD Forschung bei den Techniken des UCD und der ethnografischen Methoden.

In der PD Forschung entwickelten sich in den letzten Jahren mehrere Strömungen zur Integration von PD mit anderen Disziplinen. Unter der Bezeichnung Distributed Participatory Design (DPD) sollen PD und CSCW kombiniert werden, um PD auch für räumlich-verteilte Software-Projekte möglich zu machen und somit dem zunehmenden Trend zur globalen Software-Entwicklung Rechnung zu tragen (vgl. [Danielson et al., 2006; Danielson et al., 2008]). Die Integration von PD in das Verständnis der Information System Development Forschung (ISD) hat den Zweck, PD nicht nur als Sammelsurium von Prinzipen und Techniken zu verstehen sondern für PD auch ein Prozessmodell für Software-Projekte anzubieten (vgl. [Kawalek und Wood-Harper, 2002; Pekkola et al., 2006]). Weitere Arbeiten widmen sich dem Einsatz von PD-Techniken für den Anwender-Support in der Betriebsphase von Software (vgl. [Walldius et al., 2009]). Einen gänzlich anderen Ansatz verfolgt das Konzept des Meta-Designs (vgl. [Fischer et al., 2004]). Demnach soll bei der Entwicklung der Software bereits berücksichtigt werden, wie dem Anwender Möglichkeiten zur Konfiguration und Erweiterbarkeit angeboten

werden können, damit die Anwender die Software während des Betriebes eigenständig an ihre Bedürfnisse anpassen bzw. Funktionen zweckentfremden können.

2.7.4 Zusammenfassung der Erfolgsfaktoren der Anwenderbeteiligung

Tabelle 2.3 listet die wesentlichen Erfolgsfaktoren aus den oben genannten Paradigmen auf. Sie beschreiben relevante Rahmenbedingungen für eine erfolgreiche Anwenderbeteiligung in Software-Projekten. Allerdings stellen diese Faktoren in erster Linie abstrakte Richtlinien zur Vorgehensweise, d.h. Prinzipien, dar.

Zur Ableitung von Bewertungskriterien für Methoden zur Anwenderbeteiligung sind sie daher nicht geeignet, da sie noch operationalisiert werden müssen. Daher sind in einem weiteren Schritt empirische Studien zu Methoden zur Anwenderbeteiligung zu untersuchen und Vor- und Nachteile der Anwenderbeteiligung zusammenzufassen.

Erfolgsfaktoren	**Paradigmen**
Aktive und frühe Einbindung der Anwender über alle Projektphasen	UCD, EM, PD
Einfache und verständliche Darstellung der Entwürfe	UCD, PD
Früher und kontinuierlicher Einsatz von Prototypen	UCD, PD
Iterative und inkrementelle Entwicklungsprozesse	UCD
Analysen und Evaluationen im Nutzungskontext der Anwender	UCD, EM, PD
Multidisziplinäre Zusammenstellung der Entwickler und Anwender	UCD, EM
Einfache Anpassbarkeit der Software	UCD, PD
Entscheidungskompetenz der Anwender	UCD, PD, EM
Befähigung der Anwender zur Mitarbeit und Sicherstellung der Transparenz	PD

Tabelle 2.3: Erfolgsfaktoren der Anwenderbeteiligung aus UCD, PD und EM

2.8 Vor- und Nachteile der Anwenderbeteiligung

In den letzten drei Jahrzehnten wurden zahlreiche Studien zu Vor- und Nachteilen der Anwenderbeteiligung in Software-Projekten durchgeführt. Die Literaturlandschaft ist dabei zum einen durch positive Studienberichte über den Einsatz neuer Methoden der Anwenderbeteiligung (z.B. [Carmel et al., 1993; Dodd und Carr, 1994]) und zum anderen durch reflektierende Überblicksartikel und Metastudien (z.B. [Kujala, 2003; McKeen und Guimaraes, 1997]), die eine nüchterne und neutrale Position zum Stand der Forschung zur Anwenderbeteiligung beziehen, geprägt. Im Folgenden soll der Fokus daher auf letztere gelegt werden, um sowohl Erfolgsfaktoren als auch Risiken einer Anwenderbeteiligung zu erfassen. Hierzu werden im Rahmen einer Literaturrecherche zunächst mögliche Vorteile einer Anwenderbeteiligung aufgelistet und anschließend mögliche Nachteile erörtert. Abschließend

werden Vor- und Nachteile tabellarisch zusammengefasst und Kategorien für eine bessere Übersicht gebildet.

Bei der Literaturrecherche fällt auf, dass häufig nur zwischen den Rollen Anwender und Entwickler unterschieden und keine feinere Differenzierung zwischen Anwender, Moderator, Entscheider und Entwickler vorgenommen wird. Stattdessen werden im Entwickler die Rollen des Moderators, Entscheiders und Entwicklers vereint. Da in diesem Teil der Arbeit noch keine Differenzierung der Rollen notwendig ist, wird diese vereinfachte Form in diesem Kapitel übernommen.

Nach McKeen und Guimarares [1997] führt eine Anwenderbeteiligung möglicherweise zu genaueren und vollständigeren Informationen über Anwenderanforderungen (vgl. [Norton und McFarlan, 1975; Robey und Farrow, 1982]), mehr Wissen über das Unternehmen und die organisatorische Einheit, die von dem zukünftigen System unterstützt werden soll (vgl. [Lucas, 1974]), der Reduzierung von unnötigen Funktionen im zukünftigen System (vgl. [Robey und Farrow, 1982]), einem besserem Verständnis des zukünftigen Systems (vgl. [Lucas, 1974; Robey und Farrow, 1982]), realistischeren Erwartungen der Anwender an das entwickelte System (vgl. [Gibson, 1977]) , einer Gelegenheit für Verhandlungen und Konfliktlösungen zwischen Anwender und Entwickler (vgl. [Keen, 1981]), einer Reduzierung von Widerständen seitens der Anwender gegen das entwickelte System (vgl. [Lucas, 1974]) und einem größerem Bekenntnis zu dem zukünftigen System und seinem Erfolg (vgl. [Lucas, 1974; Markus, 1983; Robey und Farrow, 1982])).

Zudem kann sich nach Zeffane et al. [1998] eine intensive Anwenderbeteiligung positiv auf die wahrgenommene Qualität der mit dem zukünftigen Informationssystem produzierten Daten auswirken. In ihrer Befragung von 308 Managern großer australischer Unternehmen stellte sich heraus, dass u.a. Aktualität, Genauigkeit und Vollständigkeit der Daten von der Art der Anwenderbeteiligung abhängen. In den Untersuchungen von Chatzoglou und Macaulay [1996] konnte gezeigt werden, dass mit einer angemessenen Anwenderbeteiligung die Iterationen der Kommunikation zwischen Anwender und Entwickler verringert werden konnten. Sridhar et al. [2009] konnten einen positiven Einfluss von Anwenderbeteiligung auf die Planung von Informationssystemen und die Bereitschaft der Anwender zur Partizipation feststellen. Kujala [2005] konnte in ihren Feldstudien beobachten, dass eine Anwenderbeteiligung zudem die Wissenslücken der Entwickler über die Bedürfnisse der Anwender aufzeigt und die Entwickler sich dieser Lücken erst durch die Interaktion mit den Anwendern bewusst werden.

Allerdings existieren auch zahlreiche Risiken bzw. Hindernisse der Anwenderbeteiligung. Grudin [1991b] befragte über 200 Entwickler („interface designer“) nach ihren Erfahrungen mit Anwenderbeteiligung. Die Mehrheit der Befragten arbeitete in Software-Projekten der Kategorie „Produktentwicklung“ in großen Organisationen. Als größte Hindernisse der Anwenderbeteiligung wurden die Motivation der Entwickler, die Identifikation der geeigneten Anwender, die fehlende Zeit und der Zugriff auf die Anwender genannt. Außerdem war den Entwicklern unklar, wie sie die Anwender zur Mitarbeit motivieren und dabei gewinnbringend ins Projekt integrieren können.

In den Arbeiten von Wilson et al. [1996; 1997] wurden über 20 Entwickler („practitioner“, „designer“) nach Kosten und Nutzen sowie Herausforderungen und Erfolgsfaktoren der Anwenderbeteiligung in internen Entwicklungsprojekten befragt. Als Herausforderungen bzw. Hindernisse wurde beschrieben, dass Anwender nicht alle gewünschten Informationen einbringen konnten und den Entwick-

lungsprozess nicht kannten. Dabei brachten die Anwender neue Konzepte ein und produzierten eine sehr große Menge an Informationen, die zu zeitaufwändigen Diskussionen und Auswertungen führten. Außerdem hatte jeder Anwender unterschiedliche Vorstellungen und Prioritäten, so dass Kompromisse zwischen den Wünschen der Anwender gefunden werden mussten. Negativ fiel zudem auf, dass die Ansprüche der Anwender mit der Zeit stiegen und dass für eine ausreichende Repräsentativität sehr viele Anwendergruppen beteiligt werden mussten. Hierbei wurde vor allem der zeitliche Aufwand für Kontaktaufnahme und Besprechungen mit den Anwendern als zu hoch bewertet. Ähnliche Ergebnisse ergaben sich auch bei den Untersuchungen von Damoderan [1996], Hedberg [1975], Hirschheim [1989] und Kujala [2008].

Heinbokel et al. [1996] konnten in 29 langjährigen Feldstudien feststellen, dass mit dem Anstieg der Anwenderbeteiligung eine Verminderung von Projekterfolg und Innovation verbunden war und Anwenderbeteiligung zur Reduktion der Flexibilität der Entwickler führte. Darüber hinaus kam es zu Konflikten zwischen Entwicklern und Anwendern, als Anwender in einer späten Phase des Entwicklungsprojektes neue Ideen einbrachten und Änderungen forderten.

Weitere Bedenken gegen eine Anwenderbeteiligung fassen Steen et al. [2007] zusammen:

- Werden nur wenige Anwender beteiligt, kann die resultierende Software möglicherweise für die Mehrheit der zukünftigen Anwender ungeeignet sein, falls sich die Wünsche der beteiligten Anwender stark von denen der anderen unterscheiden (vgl. [Stewart und Williams, 2005]).
- Wird den Meinungen der Anwender zu stark Beachtung geschenkt, können möglicherweise bessere bzw. kreativere Ideen der Entwickler keine Anwendung finden (vgl. [Hekkert und Van Dijk, 2001; Panne et al., 2003]).

Außerdem existieren nach Weltz und Ortmann [1992] teilweise auch Kommunikationsprobleme und Interessenskonflikte zwischen Anwender und Entwickler. Anwender kennen im Regelfall die technischen und ökonomischen Hintergründe der Software-Entwicklung nicht und messen der hohen Gebrauchstauglichkeit der Software höchste Priorität bei. Entwickler kennen hingegen die Arbeitssituation der Anwender nicht. Für sie ist in erster Linie die Kosten- und Termineinhaltung wichtig. Die Gebrauchstauglichkeit muss aufgrund von Kosten- und Termindruck eine untergeordnete Rolle spielen. Nach den Erfahrungen von McKeen und Guimarares [1997] können Anwender verärgert werden, wenn ihren Vorschlägen keine Beachtung geschenkt wird. Ist die Beziehung zwischen Anwendern und Entwicklern aus historischen Gründen schon angespannt, besteht die Gefahr, dass Anwender sich absichtlich kontraproduktiv in neuen Projekten verhalten, um jede Gelegenheit zum verbalen Schlagaustausch zu nutzen. Daher empfehlen McKeen und Guimarares [1997] eine stufenweise Intensivierung der Anwenderbeteiligung abhängig vom Anspruch des Software-Projektes und von den vorhandenen Rahmenbedingungen.

Zusammenfassend lassen sich die Vor- und Nachteile in die drei Kategorien „Qualität der Anwenderbeiträge und Anforderungen“, „Effizienz der Entwickler“ und „Zufriedenheit und Einstellung der Anwender und Entwickler“ einteilen. Tabelle 2.4 ordnet die oben genannten Vor- und Nachteile diesen Kategorien stichwortartig zu. Dabei wird offensichtlich, dass eine Anwenderbeteiligung zwar zu höher Qualität der Anwenderbeiträge aber auch zu einer niedrigeren Effizienz führen kann.

Kategorie	Vorteile	Nachteile
Qualität der Anwenderbeiträge und Anforderungen	- Genauere und vollständigere Informationen über Anforderungen der Anwender - Mehr Wissen über das Unternehmen und die organisatorische Einheit, die von dem zukünftigen System unterstützt werden soll - Realistischere Erwartungen der Anwender an das entwickelte System - Reduzierung von unnötigen Funktionen im zukünftigen System	- Nichtberücksichtigung von kreativen Ideen der Entwickler - Repräsentativität der Anwender schwierig sicherzustellen
Effizienz der Entwickler	- Reduzierung der Anzahl an Iterationen zwischen Entwickler und Anwender - Gelegenheit für Verhandlungen und Konfliktlösungen zwischen Anwender und Entwickler	- Kosten- und Termineinhaltung spielt für Anwender untergeordnete Rolle - Hoher Aufwand, um Kompromisse zwischen unterschiedlichen Vorstellungen von Anwendern zu finden - Hoher Aufwand für Kontaktaufnahme und Besprechungen mit Anwendern - Zeitaufwändige Diskussionen aufgrund fehlender Kenntnisse der Anwender über Entwicklungsprozesse und Technologien - Reduktion der Flexibilität der Entwickler
Zufriedenheit und Einstellung der Anwender und Entwickler	- Steigerung der Qualität des zukünftigen Systems - Größeres Bekenntnis zu dem zukünftigen System und seinem Erfolg - Reduzierung von Widerständen der Anwender gegen das entwickelte System - Besseres Verständnis für das zukünftige System - Stärkere Nutzung des zukünftigen Systems	- Unzufriedenheit der Anwender bei Ablehnung von Anwenderbeiträgen - Austragungsort für historisch gewachsene Konflikte zwischen Anwender und Entwickler - Steigende Ansprüche der Anwender

Tabelle 2.4: Vor- und Nachteile der Anwenderbeteiligung in Software-Projekten

2.9 Bewertungskriterien für Methoden zur Anwenderbeteiligung

Für die Untersuchung von Methoden zur Anwenderbeteiligung sind nach Ives und Olson [Ives und Olson, 1984; 1981] Variablen bzw. Bewertungskriterien zu definieren, anhand derer ein eindeutiger Zusammenhang zwischen den angewendeten Methoden und den erzielten Ergebnissen hergestellt werden kann. Da in Software-Projekten eine Vielzahl an unterschiedlichen Faktoren Einfluss auf den Projekterfolg nehmen, ist nicht der Projekterfolg oder die Qualität des zukünftigen Systems als einzelner Indikator sondern eine differenzierte Betrachtung des Projektverlaufs und der Projektergebnisse maßgebend. Ives und Olson [1984] identifizierten mithilfe einer Metaanalyse von 30 empirischen Studien zu Anwenderbeteiligung mehrere Bewertungskriterien:

- Qualität des zukünftigen Systems
- Nutzung des zukünftigen Systems durch den Anwender
- Einstellung des Anwenders zum zukünftigen System
- Zufriedenheit des Anwenders mit dem zukünftigen System

Allerdings merken Ives und Olson [1984] auch an, dass die Analyse der Effekte von Methoden der Anwenderbeteiligung auf ein Software-Projekt komplex ist, da eine wissenschaftlich signifikante Berechnung der Zusammenhänge zwischen eingesetzten Methoden und gemessenen Effekten sehr schwierig durchzuführen ist. Idealerweise sollte eine ökonomische Bewertung der Methoden möglich sein, um den Effekt von alternativen Methoden auf den Projekt- bzw. Unternehmenserfolg ökonomisch beurteilen und die beste Methode mit den höchsten Gewinnchancen wählen zu können. Dies ist jedoch nicht möglich, da Kosten und Nutzen einer Methode schwierig isoliert betrachtet werden können und vergleichbare historische Daten zum Vergleich der Nutzen von Methoden im Regelfall nicht in Unternehmen aufgezeichnet werden. Daher wird im Allgemeinen die Bewertung der Qualität des zukünftigen Systems und der Akzeptanz der Anwender empfohlen, da diese einen abstrakten Zusammenhang zwischen den Effekten von Methoden zur Anwenderbeteiligung und dem Projekterfolg darstellen (vgl. [Ives und Olson, 1984]).

Die Bewertung der Qualität des zukünftigen Systems und der Akzeptanz der Anwender ist nach Ives und Olson [1984] mit zahlreichen Schwierigkeiten in der Datenerhebung verbunden. So werden Daten erst zum Start eines Projektes erhoben, so dass keine Vergleichsdaten aus vorherigen Projekten vorliegen. Außerdem wird bei der Datenerhebung im Regelfall nicht zwischen unterschiedlichen Projektphasen mit unterschiedlichen Intensitäten der Anwenderbeteiligung unterschieden. Die Zufriedenheit der Anwender ist stark von weiteren Rahmenbedingungen abhängig und kann daher nicht direkt auf die Qualität eines Systems zurückgeführt werden. Außerdem kann die Nutzung eines Systems kein verlässlicher Indikator sein, wenn das entwickelte System elementar bzw. verpflichtend für die Arbeit der Anwender ist. Dies soll nicht grundlegend eine wissenschaftliche Betrachtung der Anwenderbeteiligung in Frage stellen, aber zu einer sorgfältigen Vorgehensweise, einem kritischen Umgang mit Studienergebnissen und der Entwicklung besserer Untersuchungsmethoden motivieren.

Dieser Forderung nachkommend, stellen u.a. Baroudi et al. [1986], McKeen und Guimamaes [1997], Galetta und Lederer [1989], Kappelmann und McLean [1992; 1995] sowie He und King [1995; 2008], einen direkten Zusammenhang zwischen „Zufriedenheit der Anwender mit dem zukünftigen System“ („user satisfaction“), „Einstellung der Anwender zum zukünftigen System“ („user attitude“), „Benutzung des zukünftigen Systems“ („system usage“) und „Qualität des zukünftigen Systems“ („system quality“) her. Die Einstellung der Anwender gegenüber dem zukünftigen System bzw. dem Software-Projekt beschreibt, welche subjektive Haltung der Anwender gegenüber einem System bzw. einem Software-Projekt einnimmt (vgl. [Barki und Hartwick, 1994; Fishbein und Ajzen, 1975]). Sie wirkt sich zur Projektlaufzeit auf die Motivation und Arbeitsmentalität der Anwender aus, an einem Software-Projekt teilzunehmen und konstruktive Beiträge einzubringen. Abhängig von der Qualität des zukünftigen Systems und von der Einstellung der Anwender im Software-Projekt ergibt sich schließlich die Zufriedenheit der Anwender, und darauf aufbauend die Indikation für die spätere Benutzung des Systems durch die Anwender. Die Einstellung der Anwender drückt sich demnach u.a. durch ihre Zufriedenheit mit der Methode zur Anwenderbeteiligung und ihrer Zufriedenheit mit dem Verlauf des Projektes aus.

Eine umfassendere Sichtweise entwickelte Kujala [2003; 2008] mithilfe ihres Literaturreviews und ihren Feldstudien zu Vor- und Nachteilen einer Anwenderbeteiligung in Software-Projekten, bei der sie direkte Zusammenhänge zwischen Anwenderbeteiligung, Effizienz der Entwicklungsteams, Zufriedenheit der Anwender sowie der Qualität der Anforderungen und des Systems ableiten konnte (vgl. Abbildung 1.1, S. 2). Allerdings schweigt sie sich darüber aus, wie diese Auswirkungen konkret zu messen bzw. bewerten sind.

Einen anderen Weg schlugen Carrizo et al. [2008], Davis et al. [2006], Dieste et al. [2008], Dieste und Juristo [2010] und Hickey und Davis [2003] ein. Sie erarbeiteten im Rahmen von Expertenbefragungen und Literaturreviews Kriterien, die Praktiker bei der Wahl von Methoden zur Anforderungserhebung unterstützen sollen. Ausgehend von den vorgegebenen Rahmenbedingungen und der Zielsetzung eines Software-Projekts sollen geeignete Methoden anhand eines Kriterienkatalogs identifiziert werden können. Es finden sich bei ihnen jedoch ebenfalls keine Kriterien um Methoden bewerten zu können. Sie enthalten stattdessen Aussagen darüber, für welchen Zweck welche Methode geeignet sein kann, ohne jedoch auf ihre möglichen Auswirkungen einzugehen.

Zusammengefasst ist festzuhalten, dass in der Literatur bislang keine konkreten Bewertungskriterien zur Bewertung der Potentiale und Grenzen von Methoden zur Anwenderbeteiligung existieren. Daher wird im Folgenden ein Modell über die Auswirkungen von Methoden zur Anwenderbeteiligung in Software-Projekten entwickelt und darauf aufbauend Bewertungskriterien abgeleitet. Grundlage des Modells bildet die Kategorisierung der möglichen Vor- und Nachteile der Anwenderbeteiligung aus Kapitel 2.8 und die darauf aufbauende Schlussfolgerung, dass Methoden zur Anwenderbeteiligung in direktem Zusammenhang mit der Qualität der Anwenderbeiträge und Anforderungen, der Effizienz der Anwender und Entwickler sowie der Zufriedenheit der Anwender und Entwickler steht, wie es bereits auch von Kujala [2008] in ähnlicher Weise empfohlen wird.

Der Einfluss von Methoden zur Anwenderbeteiligung auf die Akzeptanz der Anwender kann durch die Grad der Belastung der Anwender und das erreichte Ergebnis definiert werden (vgl. [Sarodnick und Brau, 2006]). Idealerweise möchten Anwender mit einem geringen Aufwand einen maximalen Effekt erreichen. Aus Sicht des Anwenders ist somit die Effizienz ihrer Beteiligung ohne eine Überforderung

sowie die Qualität des zukünftigen Systems entscheidend. Sie wirkt sich zur Projektlaufzeit auf die Motivation und Arbeitsmentalität der Anwender aus, an einem Software-Projekt teilzunehmen und konstruktive Beiträge einzubringen. Die Belastung der Anwender drückt sich durch die Effizienz und Komplexität zur Erstellung ihrer Beiträge aus. Abhängig von der Qualität des zukünftigen Systems und von der Belastung der Anwender im Software-Projekt ergibt sich schließlich die Zufriedenheit der Anwender, und darauf aufbauend die Indikation für die spätere Benutzung des Systems durch die Anwender.

Eine Anwenderbeteiligung kann zur Verbesserung der Systemqualität führen, indem die Qualität der Anforderungen verbessert wird (vgl. [Kujala, 2008]). Allerdings wird im Rahmen der Anforderungsanalyse eines Software-Projekts auch eine Vielzahl an Anforderungen erhoben, die keinen direkten Bezug zum Anwender haben (z.B. nicht-funktionale Anforderungen zu Skalierbarkeit, Flexibilität). Außerdem erfolgen nach der Interaktion mit Anwendern weitere Entscheidungsprozesse, die unabhängig von der Methode zur Anwenderbeteiligung (vgl. Kapitel 2.2 und 2.3) zu der finalen Zusammenstellung der Anforderungen führen. Die Bewertung der Qualität der Anforderungen reicht somit nicht als Indikator für Methoden zur Anwenderbeteiligung aus. Stattdessen bietet sich die Bewertung der Qualität der eingebrachten Anwenderbeiträge als Indikator einer Methode zur Anwenderbeteiligung an, da sie sich direkt auf eine Methode zur Anwenderbeteiligung anwenden lässt. Da im Unterschied zu Anforderungen bzw. Anforderungsdokumenten (vgl. Kapitel 2.5) in der Literatur keine standardisierten Bewertungskriterien für Anwenderbeiträge existieren, wird zunächst für diese Arbeit die Annahme getroffen, dass folgende Bewertungskriterien Anwendung finden sollten:

- Verständlichkeit: Die Anwenderbeiträge sollten in erster Linie schnell und einfach nachvollzogen werden können.
- Konstruktivität: Die Anwenderbeiträge enthalten konstruktive Hinweise auf die Bedürfnisse der Anwender und auf Gestaltungsmöglichkeiten des Systems.
- Umfang: Die Anwenderbeiträge enthalten alle für den Entwickler notwendigen Informationen, so dass keine Rückfragen des Entwicklers notwendig werden.
- Anzahl: Die Anzahl der Anwenderbeiträge ist ein Indikator dafür, wie stark die Kreativität und Motivation der Anwender gefördert wird.
- Repräsentativität: Die Anwenderbeiträge sollten von unterschiedlichen Anwendern stammen, um die unterschiedlichen Meinungen und Vorstellungen erfassen zu können.
- Kontext-Bezug: Die Anwenderbeiträge enthalten Informationen über den Nutzungskontext der Anwender.

Die Effizienz der Entwicklung drückt sich unmittelbar durch ihre Kosten aus. Eine Anwenderbeteiligung kann Projektlaufeit, Fehleranzahl und Kosten eines Software-Projekts senken (vgl. [Chatzoglou und Macaulay, 1996]). Sie kann aber auch zu häufigeren Iterationen und Verzögerungen führen (vgl. [Heinbokel et al., 1996]). Diese Auswirkungen können jedoch nicht eindeutig von weiteren Faktoren wie z.B. Abneigungen zwischen Entwickler und Anwender, Komplexität des Systems, etc. getrennt werden. Für die isolierte Bewertung einer Methode zur Anwenderbeteiligung kann der Aufwand für die Vor- und Nachbereitung der Anwenderbeteiligung sowie für die Interaktion zwischen Anwender und Entwickler herangezogen werden.

Somit ergeben sich folgende Auswirkungen durch Methoden zur Anwenderbeteiligung:

- Eine Methode zur Anwenderbeteiligung wirkt sich direkt auf die Effizienz der Entwicklung, die Qualität der Anwenderbeiträge und die Belastung der Anwender aus.

- Die Effizienz der Entwicklung bemisst sich im Aufwand für die Vor- und Nachbereitung der Anwenderbeteiligung sowie für die tatsächliche Interaktion mit dem Anwender. Zudem ist die Anzahl der notwendigen Iterationen ein Indikator, wie häufig eine Kommunikation zwischen Anwender und Entwickler notwendig war. Sie hat Einfluss auf die Belastung der Entwickler.

- Die Qualität der Anwenderbeiträge wird durch die Verständlichkeit, die Konstruktivität, den Umfang, die Anzahl, die Repräsentativität und den Kontext-Bezug der Anwenderbeiträge ausgedrückt. Sie trägt zur Effizienz der Entwicklung und die Qualität der Anforderungen bei und beeinflusst auch die Belastung der Entwickler.

- Die Belastung der Anwender beschreibt ihre Beanspruchung durch ihre Beteiligung im Software-Projekt. Eine Methode nimmt dabei Einfluss auf die Effizienz und Komplexität zur Erstellung der Anwenderbeiträge. Sie trägt entscheidend dazu bei, wie konstruktiv Anwender sich einbringen können und wirkt sich somit auch auf die Qualität der Anwenderbeiträge aus.

- Die Belastung der Entwickler ergibt sich aus der Effizienz der Entwicklung und der Qualität der Anwenderbeiträge. Sie bestimmt ihre Motivation zur Interaktion mit den Anwendern und die Bereitschaft zur Akzeptanz von Anwenderbeiträgen bzw. den Anwenderbeiträgen. Sie prägt schließlich die Zufriedenheit der Entwickler.

- Die Qualität der Anforderungen lässt sich nach Davis et al. [1997] durch ihre Verständlichkeit, ihre Vollständigkeit, ihre Korrektheit und ihre Verifizierbarkeit beurteilen. Sie nimmt maßgeblich Einfluss auf die Qualität des (zukünftigen) Systems.

- Die Qualität des Systems drückt sich durch ihre Gebrauchstauglichkeit aus. Da sie das Resultat der Bemühungen der Anwender und Entwickler darstellt, ist sie maßgebend für ihre Zufriedenheit. Außerdem ist von der Qualität des Systems auch das spätere Nutzungsverhalten der Anwender abhängig.

- Die Zufriedenheit der Anwender und Entwickler ergibt sich aus ihrer Zufriedenheit mit der Qualität des Systems und mit ihrer Belastung im Verlauf des Projekts. Für den Anwender spielt zudem ihre Zufriedenheit auch eine Rolle bei der Bereitschaft zur späteren Benutzung des Systems. Aus ihrer Zufriedenheit kann möglicherweise auch auf die zukünftige Form der Anwenderbeteiligung in Nachfolgeprojekten geschlossen werden.

Abbildung 2.2 illustriert die Zusammenhänge der oben genannten Auswirkungen. Die Pfeile symbolisieren dabei die Richtung der Beeinflussung. Dabei kann ein Pfeil sowohl eine positive als auch eine negative Wirkung symbolisieren. Da die Wahl der Methode offen ist, kann sich somit die Wahl der Methode entweder positiv oder negativ auf die einzelnen Parameter auswirken. Durch die weitere Verkettung werden die negativen bzw. positiven Auswirkungen weitergegeben. Beispielsweise kann sich eine Methode zur Anwenderbeteiligung negativ auf die Belastung der Anwender auswirken. Dies

wiederum wirkt sich negativ auf die Zufriedenheit der Anwender und auf die Qualität der Anwenderbeiträge aus.

Im Vergleich zu Kujala [2008] unterscheidet dieses Modell zwischen Qualität von Anwenderbeiträgen und Qualität von Anforderungen sowie zwischen Belastung und Zufriedenheit der Anwender. Während Anforderungen das Ergebnis der gesamten Anforderungsanalyse darstellen, sind Anwenderbeiträge direkte Ergebnisse einer Methode zur Anwenderbeteiligung und daher für die Bewertung einer Methode von großer Relevanz. Die Zufriedenheit der Anwender hängt nicht nur von der Qualität des Systems sondern auch vom Ablauf der Anwenderbeteiligung ab. Dies findet sich in der Belastung der Anwender während der Anwenderbeteiligung im Software-Projekt wieder. Die Belastung der Anwender kann sich zudem auch auf die Qualität der Anwenderbeiträge auswirken, falls die Anwender zu stark beansprucht werden. Außerdem kommt die Belastung der Entwickler als Ergebnis der Effizienz der Entwicklung und der Qualität des Systems neu hinzu. Sie drückt aus, ob von Seiten der Entwickler eine Anwenderbeteiligung als hilfreich oder belastend angesehen wird und eine Methode zur Anwenderbeteiligung für zukünftige Anwendungen Akzeptanz finden wird.

Für die Bewertung einer Methode zur Anwenderbeteiligung lässt sich aus diesem Modell schließen, dass primär die Effizienz der Entwicklung, die Qualität der Anwenderbeiträge und die Belastung der Anwender zu erfassen ist. Für die Ausprägung der Bewertungskriterien wird eine drei-stufige Ordinalskala („Niedrig“ < „Mittel“ < „Hoch“) in Anlehnung an die Kategorisierung von Carrizo [2008] angewendet. Tabelle 2.5 fasst die Bewertungskriterien an Methoden zur Anwenderbeteiligung zusammen.

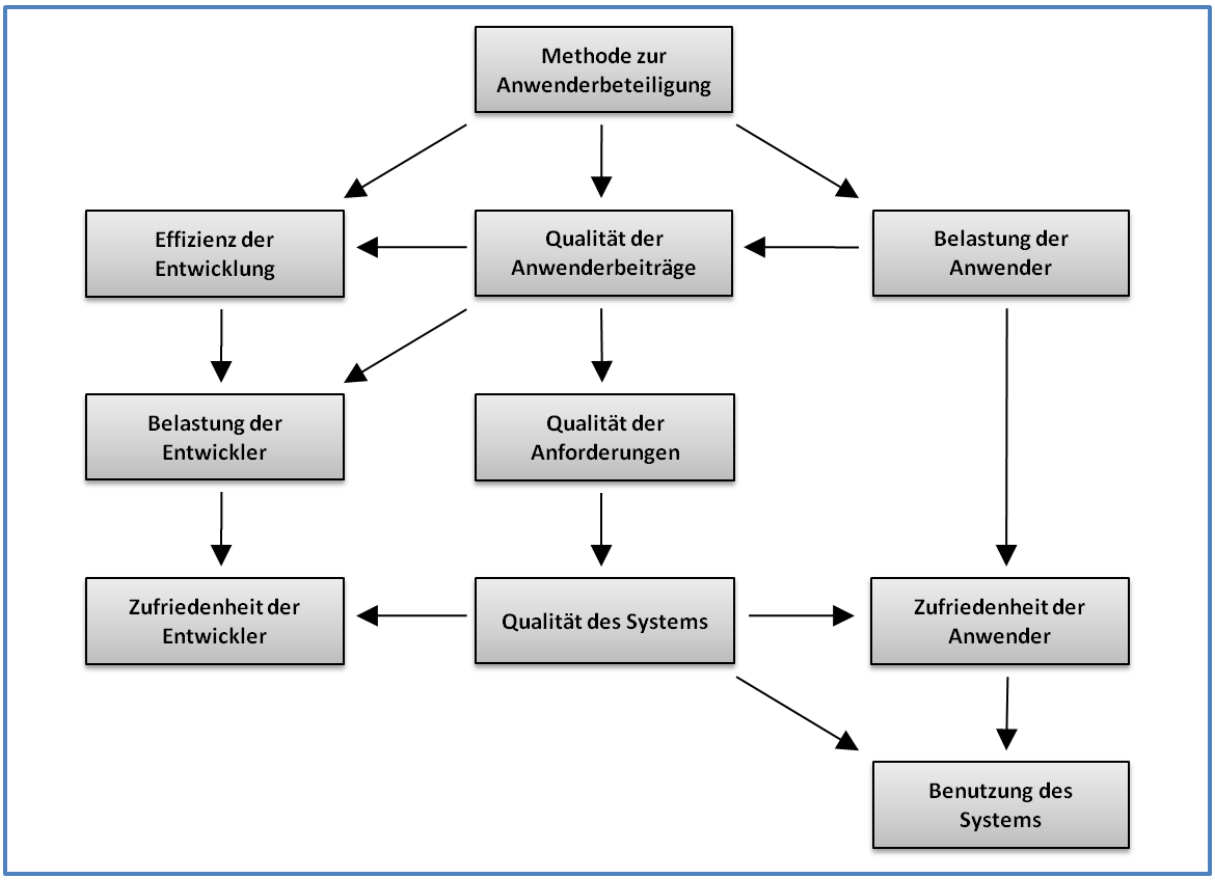

Abbildung 2.2: Auswirkungen von Methoden zur Anwenderbeteiligung in Software-Projekten

Parameter	Bewertungskriterien	Ausprägung
Effizienz der Entwicklung	- Effizienz der Vorbereitung	Niedrig/ Mittel/ Hoch
	- Effizienz der Interaktion mit Anwender	Niedrig/ Mittel/ Hoch
	- Effizienz der Nachbereitung	Niedrig/ Mittel/ Hoch
Qualität der Anwenderbeiträge	- Verständlichkeit der Anwenderbeiträge	Niedrig/ Mittel/ Hoch
	- Konstruktivität der Anwenderbeiträge	Niedrig/ Mittel/ Hoch
	- Umfang der Anwenderbeiträge	Niedrig/ Mittel/ Hoch
	- Anzahl der Anwenderbeiträge	Niedrig/ Mittel/ Hoch
	- Repräsentativität der Anwenderbeiträge	Niedrig/ Mittel/ Hoch
	- Kontext-Bezug der Anwenderbeiträge	Niedrig/ Mittel/ Hoch
Belastung der Anwender	- Effizienz der Erstellung von Anwenderbeiträgen	Niedrig/ Mittel/ Hoch
	- Komplexität der Erstellung von Anwenderbeiträgen	Niedrig/ Mittel/ Hoch

Tabelle 2.5: Bewertungskriterien für Methoden zur Anwenderbeteiligung

2.10 Fazit zu Kapitel 2

Ziel von Kapitel 2 war es, Bewertungskriterien an Methoden zur Anwenderbeteiligung zusammenzustellen. Mit den Bewertungskriterien sollen die Potentiale der unterschiedlichen Methoden zur Anwenderbeteiligung untersucht und Verbesserungsmöglichkeiten identifiziert werden können. Daher war ein grundlegendes Verständnis zu Methoden zur Anwenderbeteiligung und zu ihren möglichen Auswirkungen auf Verlauf und Ergebnisse von Software-Projekten aufzubauen.

Hierfür wurden zugrundeliegende Begriffe zu Software bzw. System, Software-Projekt, Vorgehensmodelle, Anforderungsdokumente und Anwenderbeteiligung aufgearbeitet und der aktuelle Stand der Forschung zu Anwenderbeteiligung in Software-Projekten wiedergegeben. Darauf aufbauend wurden Vor- und Nachteile der Anwenderbeteiligung erörtert und ein Modell zu den Auswirkungen von Methoden zur Anwenderbeteiligung entwickelt. Dies mündete schließlich in eine Auflistung von Bewertungskriterien für Methoden zur Anwenderbeteiligung.

Dabei konnten folgende eigene Beiträge herausgearbeitet werden:

- Die Analyse der Beschreibungen der Rollen in Software-Projekten aus etablierten Vorgehensmodellen ergab, dass die bisherigen Definitionen der Rollen keinen direkten Bezug zu Aufgaben und Verantwortlichkeiten im Rahmen einer Anwenderbeteiligung enthalten. Mithilfe der Literaturrecherche zu Rollen in der Anwenderbeteiligung konnten die Rollen Anwender, Entwickler, Entscheider und Moderator als die relevanten Beteiligten in einer Anwenderbeteiligung in Software-Projekten definiert werden, womit eine einheitliche und vereinfachte Definition der relevanten Rollen gegeben werden konnte. Für jede dieser Rolle wurde der Bezug zu den üblichen Rollen in Vorgehensmodellen hergestellt, Aufgaben, Ver-

antwortlichkeiten und Bedürfnisse zugeordnet und somit die Machbarkeit der gewählten Einteilung aufgezeigt. Für diese Arbeit vereinfacht diese Rollendefinition die Begrifflichkeit und erleichtert die Analyse von Anforderungen der Rollen an Methoden zur Anwenderbeteiligung.

- Aus der Literaturrecherche zu UCD, EM und PD wurden relevante Erfolgsfaktoren der Anwenderbeteiligung herausgearbeitet, die in dieser Übersicht bislang nicht in der Literatur vorlagen. Für diese Arbeit stellen sie die Grundlage für die weiteren Arbeiten dar und werden als Prinzipien zur Gestaltung von Methoden zur Anwenderbeteiligung angewendet. Allerdings eigenen sich die Erfolgsfaktoren nicht zur Ableitung von Bewertungskriterien für Methoden zur Anwenderbeteiligung, da sie sich auf Prinzipien und nicht auf Methoden beziehen.

- Die Literaturrecherche zu Vor- und Nachteilen der Anwenderbeteiligung ergab, dass sich eine Anwenderbeteiligung auf die Qualität der Anwenderbeiträge und Anforderungen, auf die Effizienz der Entwickler und auf die Zufriedenheit und Belastung der Anwender und Entwickler auswirken kann. Dabei zeigte sich in den bisherigen Studien, dass sich eine Anwenderbeteiligung positiv auf die Qualität der Anforderungen und die Zufriedenheit der Anwender und somit auf die Qualität des Systems auswirken kann. Allerdings kann die Anwenderbeteiligung die Effizienz der Entwickler senken und zu höheren Aufwänden für Kommunikation und Entscheidungsfindung führen.

- Darauf aufbauend wurde der Frage nachgegangen, wie sich anhand der ermittelten Vor- und Nachteile Bewertungskriterien für Methoden zur Anwenderbeteiligung ableiten lassen. Hierfür wurde ein Modell über die Auswirkungen von Methoden zur Anwenderbeteiligung in Software-Projekten entwickelt, das die komplexen Zusammenhänge zwischen der Wahl der Methode und dem Erfolg eines Projektes beschreibt. In Anlehnung an die Vorarbeiten von Kujala [2008], Dieste et al. [2008], Carrizo et al. [2008], Hickey und Davis [2003] und Davis et al. [2006] wurden Bewertungskriterien zu den Parametern „Qualität der Anwenderbeiträge“, „Effizienz der Entwicklung“ und „Belastung der Anwender“ operationalisiert, da diese unmittelbar durch die Wahl der Methode zur Anwenderbeteiligung beeinflusst werden.

In diesem Kapitel wurde somit die erste Forschungsfrage dieser Arbeit nach den Bewertungskriterien für Methoden zur Anwenderbeteiligung beantwortet.

Mithilfe der entwickelten Bewertungskriterien werden im zweiten Teil der Arbeit im folgenden Kapitel 3 die Potentiale und Grenzen der synchronen Anwenderbeteiligung untersucht und die Notwendigkeit von asynchronen Methoden zur Anwenderbeteiligung motiviert.

3 Potentiale und Grenzen von Methoden zur synchronen Anwenderbeteiligung

Ziel dieses Kapitels ist die Untersuchung der Potentiale und Grenzen von Methoden zur synchronen Anwenderbeteiligung. Hierzu werden zu Beginn ein Überblick über Methoden zur synchronen Anwenderbeteiligung gegeben und ihre Potentiale und Grenzen aufbauend auf der aktuellen Studienlage und anhand der Bewertungskriterien aus Kapitel 2.9 bewertet.

3.1 Überblick über Methoden zur synchronen Anwenderbeteiligung

Tabelle 3.1 listet die etablierten Methoden zur synchronen Anwenderbeteiligung in Software-Projekten auf (vgl. [Allen et al., 1993; Carrizo et al., 2008; Sarodnick und Brau, 2006]). Hierfür wurden im Rahmen einer Literaturanalyse diejenigen Methoden identifiziert, die eine Vorgehensweise zur Interaktion mit Anwendern beschreiben, in mehreren unterschiedlichen Literaturquellen erwähnt werden und eine synchrone Kommunikation, also eine zeitgleiche Anwesenheit von Anwendern und Moderatoren, erfordern.

In „Zukunftswerkstätten“ sollen neue Ideen für die zukünftige Organisation und Gestaltung des Arbeitsplatzes erarbeitet und ein realisierbarer Umsetzungsplan bestimmt werden. Unter der Zukunftswerkstatt versteht man nach Apel et al. [1998, S. 282] „*eine Methode, die unter Einbezug von Moderatoren/innen die Selbstorganisation, Wahrnehmungsfähigkeit, Fantasie und Handlungskompetenz der Teilnehmenden fördert und Möglichkeiten zur Realisierung gemeinsamer Ideen entwickeln hilft und in der Umsetzung beratend begleitet*“. „Workshops“ dienen im Allgemeinen der Diskussion von Wünschen und Vorstellungen der Anwender. Hierfür werden in einer moderierten Diskussionsrunde mit einer Gruppe von Anwendern Prototypen getestet und allgemeine Anforderungen diskutiert erfasst. Die größte Entscheidungskompetenz besitzen die Anwender, wenn sie im „Kernteam“ eines Software-Projektes mitarbeiten, den gesamten Kontext der Entwicklung miterleben und bei Entscheidungen im Entwurf und Umsetzungen anwesend sind. Dabei befindet sich der Anwender im Regelfall über die gesamte Laufzeit des Software-Projektes am gleichen Ort wie die Entwickler und nimmt an allen relevanten Besprechungen teil. Beim „Usability Test“ testet im Regelfall ein Anwender einen Prototypen unter Anleitung eines Testleiters und beschreibt seine Eindrücke und Erwartungen. Im „Living Lab“ können Anwender Systeme in einer in einer der Realität nachempfundenen Umgebung testen und kommentieren. Die Methoden „Interview“, „Beobachten“ bzw. „Contextual Inquiry“, „Teilnehmende Beobachtung“ und „Aufgabenanalyse“ dienen dazu, die Arbeitsumgebung der Anwender zu verstehen und Bedürfnisse und Probleme zu erfassen.

Methode	**Kurzbeschreibung**

Zukunfts-werkstätten	Anwender erarbeiten in einer Gruppe unter Moderation Wunschszenarios über ihre zukünftige Arbeitsgestaltung.
(Usability) Workshops	In Arbeitsgruppen mit Moderatoren, Entwickler und Entscheider werden Anforderungen und Gestaltungsmöglichkeiten diskutiert.
Kernteam	Anwender arbeiten im Team des Software-Projekts aktiv mit.
(Kooperatives) Prototyping	Anwender werden gebeten, Prototypen zu bewerten und Verbesserungsvorschläge einzubringen.
Usability Test/ Living Lab	In einer der realen Arbeitsumgebung nachempfundenen Umgebung können Anwender Software testen und Verbesserungsvorschläge einbringen.
Interview	Anwender geben ihre Meinung im Rahmen eines Gesprächs ab. Die Fragestellungen können an den Verlauf des Gesprächs angepasst werden.
Beobachten/ Contextual Inquiry	Die Anwender werden bei ihren Arbeitsprozessen in ihrer Arbeitsumgebung beobachtet. Der Moderator stellt dabei Fragen.
Teilnehmende Beobachtung	Beobachter begleiten die Anwender im Arbeitsumfeld und übernehmen auch selbst die Aufgaben der Anwender, um deren Situation nachzuvollziehen.
Soziotechnischer Walkthrough	Konzepte (z.B. Prototypen) werden den Anwendern vorgestellt und mögliche Auswirkungen auf den Arbeitskontext diskutiert.
Card Sorting	Auf Karten aus Papier werden Begriffe notiert. Eine Gruppe von Anwendern sortiert diese Begriffe in Begriffsklassen.
Brainstorming	Zu einem vorgegebenen Thema bringen Anwender in einer Gesprächsrunde ihre Lösungsideen spontan ein.
Nominal Group Technik	Zu einem Thema werden in einer moderierten Diskussion schriftlich Vorschläge von Anwendern erfasst und priorisiert.
Repertory Grid	Anwender ordnen eine vorgegebene Menge an Elementen zu einer Menge an Konstrukten und drücken somit ihre Interpretation einer Begriffswelt aus.
Szenarien/ Persona/ Anwendungsfälle	Anwender beschreiben ihre Anforderungen an Software nach vorgegebenen Strukturen in Form von Szenarien, Persona bzw. Anwendungsfällen.
Anwendergruppen	Anwender schließen sich zu Anwendergruppen zusammen, um über strategische Änderungen an der Software zu beratschlagen.
Aufgabenanalyse	Das Aufgabenumfeld der Anwender wird intensiv analysiert und eine sinnvolle Aufgabenteilung zwischen Software bzw. System und Anwender erarbeitet.

Tabelle 3.1: Methoden zur synchronen Anwenderbeteiligung in Software-Projekten

Zur kreativen Ideenfindung dienen die Methoden „Brainstorming“ und die „Nominal-Group Technik“. Moderierte Diskussionen können mithilfe der Methoden „soziotechnischer Walkthrough“, „Fokusgruppen“ und „Anwendergruppen“ erzielt werden.

Die Methoden „Card Sorting“ und „Repertory Grid“ ermöglichen die Analyse des Interpretationsraums der Anwender. Diese Methoden sind vor allem dann hilfreich, wenn Navigationsstrukturen einer Software erstellt werden sollen. Mit „Szenarien / Persona/ Anwendungsfälle“ beschreiben Anwender relevante Situationen bzw. relevante Zielgruppen, die durch eine Software unterstützt werden soll. Diese Modelle dienen als Entscheidungsgrundlage im weiteren Verlauf des Software-Projekts, so dass keine weiteren Iterationen mit Anwendern notwendig werden. Bei Szenarien werden „Geschichten“ bzw. „Storyboards“ beschrieben, die besonders relevant oder exemplarisch sind. Mit „Persona“ werden Eigenschaften und Wünsche von Stellvertretern der Zielgruppe, sogenannten Archetypen bzw. Stereotypen, ausführlich beschrieben, so dass Entwickler sich diese Personen plastisch vorstellen können. „Anwendungsfälle“ sind eine Bündelung von allen möglichen Szenarien, in die ein Anwender bei der Erreichung seiner Ziele gelangen kann.

Beim „Prototyping“ bewerten die Anwender einen oder mehrere Prototypen und drücken ihre Vorstellungen direkt am konkreten Prototypen aus. Es stellt einen Spezialfall dar, da es zwar häufig als eigenständige Methode genannt wird, aber üblicherweise ein Basiselement für andere Methoden darstellen kann. Im Unterschied zu anderen Methoden steht beim Protopying die Arbeit an Prototypen im Fokus der Entwicklung und ist zentrales Ergebnisartefakt eines Software-Projektes.

Allen Methoden zur synchronen Anwenderbeteiligung ist gemeinsam, dass für die Zusammenarbeit zwischen Anwender und Moderator gemeinsame Treffen zur Abstimmung bzw. zum Informationsaustausch organisiert werden. Der Moderator bereitet die zu besprechenden Inhalte vor und bespricht diese mit dem Anwender. Außerdem beobachtet der Moderator den Anwender bzw. alle Auffälligkeiten während des Treffens und protokolliert das Treffen. Für die weiteren Arbeitsschritte zwischen Moderator, Entwickler und Entscheider werden die Ergebnisse der Treffen schriftlich fixiert. Die Methoden unterscheiden sich u.a. in ihrer Zielsetzung, in der Intensität der Interaktion mit dem Anwender, in der Struktur der Dokumentation und in den verwendeten Kommunikationsartefakten.

Bei den synchronen Methoden werden Werkzeuge im Allgemeinen zur schriftlichen Kommunikation und zur Protokollierung der Ergebnisse verwendet. Beispielsweise können beim Brainstorming Mindmaps (vgl. [Buzan, 2006]), beim Beobachten Foto- und Videoaufzeichnungen (vgl. [Lamnek, 2005]), bei Interviews Audio-Aufzeichnungen (vgl. [Mayring, 2002]) und für Szenarios Textverarbeitungsprogramme oder Video-Aufzeichnungen eingesetzt werden.

Für Usability Tests ist der Einsatz von Werkzeugen besonders ausgeprägt. An einem speziell eingerichteten Computer kann der Anwender die Software testen und Kommentare nach der „thinking aloud“ Methode abgeben. Die „thinking aloud“ Methode sieht vor, dass der Anwender während des Testes laut denkt, also seine Eindrücke und Gedanken laut ausspricht. Dabei wird das Gesicht des Anwenders von einer Videokamera aufgenommen und der Ton über ein Mikrofon aufgezeichnet. Gleichzeitig wird die Anwendungsoberfläche des Computers über ein sogenanntes „Screenrecording“ bzw. Bildschirmaufnahme aufgenommen. Dies stellt sicher, dass die Interaktion des Benutzers mit der Software mit den Reaktionen und den gesprochenen Kommentaren des Anwenders synchronisiert werden kann. Für den Moderator und den Anwender stellt dies eine Erleichterung dar, da sie

sich vollständig auf den Usability Test konzentrieren können und der Moderator nach dem Test die Analyse der Videoaufnahmen in aller Ruhe durchführen kann. Es ist nachgewiesen, dass dadurch im Vergleich zum reinen Beobachten und Notizenerstellen mehr Details festgehalten werden können. Um die Videoanalyse zu erleichtern, bieten zahlreiche Videoanalyse-Software-Produkte die Möglichkeit, während des Testes über Tastaturkürzel Markierungen in der Videoaufzeichnung zu platzieren. So kann der Moderator bestimmte Stellen im Video mit Kategorien markieren. Mithilfe der Markierungen können die Videos nachträglich schneller analysiert und zudem die Häufigkeiten der einzelnen Kategorien ausgewertet werden. Basierend auf der Analyse erstellt der Moderator eine Liste von Verbesserungsvorschlägen. Das „Eyetracking“ ist eine Ergänzung zur Videoanalyse. Dabei werden über ein spezielles Videokamerasystem zusätzlich die Augenbewegungen des Anwenders beobachtet. So kann festgestellt werden, auf welche Stellen in der Anwendung der Anwender seine Blicke richtet.

Ein Usability-Test kann mithilfe computergestützter Werkzeuge auch räumlich verteilt erfolgen. Hierzu hat sich der Einsatz von Video-Konferenzsystemen (vgl. [Andreasen et al., 2007; Burger et al., 2008; Hartson et al., 1996]) als hilfreich erwiesen. Hierbei werden, wie in Abbildung 3.1 beispielhaft an dem Werkzeug vitero dargestellt, Anwender zu einer webbasierten Video-Telefonkonferenz eingeladen. Die Moderatoren schalten die zu testende Software per Videoübertragung zu und steuern den Ablauf. Die Anwender können die Software testen und ihre Eindrücke u.a. mit Chat-Funktionen, Zeichnungsfunktionen, mit der „thinking aloud“ Methode und per Webcam vermitteln. Wahlweise können ein oder mehrere Anwender gleichzeitig eingebunden werden. Die Ergebnisqualität von räumlich verteilten Usability-Tests ist vergleichbar mit normalen ortsgebundenen Usability-Tests im Labor (vgl. [Andreasen et al., 2007; Andrzejczak und Liu, 2010; Tullis et al., 2002]).

Abbildung 3.1: vitero Besprechungstisch [Burger et al., 2008]

Ein Beispiel für einen video-basierten Ansatz zur synchronen Anwenderbeteiligung stellt Software Cinema von Bruegge und Creighton et al. [2004; 2006] dar. Mit dem dazugehörigen Software Cinema Toolkit (vgl. Abbildung 3.2) werden Videos erstellt und annotiert, um dem Anwender und Entwickler die gewünschte Funktionsweise eines Systems zu präsentieren und zu diskutieren. Ziel des Ansatzes ist es, die objektorientierte Sicht der Entwickler auf die funktionsorientierte Sicht der Anwender abzubilden und dadurch die Kommunikation zwischen Entwicklern und Anwendern zu erleichtern. In einer frühen Phase der Software-Entwicklung eignet sich die Aufzeichnung des Ist-Zustands der Software-Nutzung bzw. der betroffenen Prozesse, während zu einem späteren Zeitpunkt die erarbeiteten Anwendungsfälle in Form eines nachgestellten Szenarios aufgezeichnet werden.

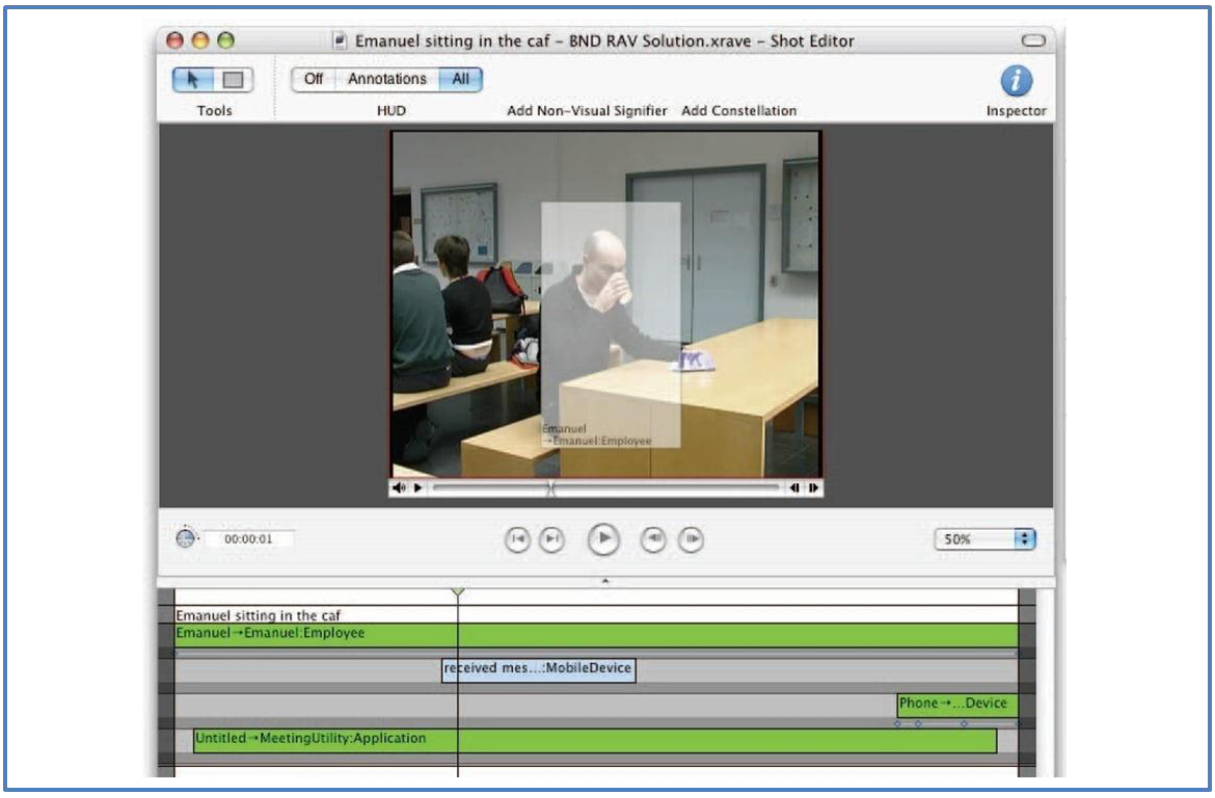

Abbildung 3.2: Software Cinema Shot Editor [Creighton et al., 2006]

3.2 Bewertung von Methoden zur synchronen Anwenderbeteiligung

Methoden zur synchronen Anwenderbeteiligung besitzen unterschiedliche Ausprägungen, da jede Methode für sich eine bestimmte Zielsetzung verfolgt und daher auch zweckorientiert darauf ausgerichtet ist. Dennoch lassen sie sich durch folgende gemeinsame Eigenschaften charakterisieren:

- Die Kommunikationspartner können unmittelbar auf die Beiträge reagieren und Sachverhalte gesammelt zu einem Zeitpunkt klären (vgl. [Schwabe et al., 2001]).
- Im persönlichen Gespräch ist das volle Spektrum der Kommunikationskanäle (z.B. Körpersprache, Tonfall) vorhanden. Dies ist für die soziale Komponente der Kommunikation und der Vermeidung von Missverständnissen von großer Bedeutung (vgl. [Schulmeister, 2006;

Schwabe et al., 2001]). Außerdem können sich Moderatoren und Entwickler nach Kujala [2003] ein Bild von den Anwendern machen – sie bleiben dadurch keine „große gesichtslose Masse“ („big faceless mass“).

- Mithilfe von technischen Kommunikationswerkzeugen ist eine ortsunabhängige räumlich-verteilte Kommunikation (z.B. über Videokonferenz oder Telefon) möglich. Hierdurch kann die Kommunikation auch persistent gespeichert und später nachgeschlagen bzw. für die weitere Kommunikation und Dokumentation übernommen werden (vgl. [Hew und Cheung, 2003]). Dies hat jedoch auch eine Reduktion der Kommunikationskanäle zur Folge (vgl. [Schulmeister, 2006]).

- Gemeinsame synchrone Arbeitstreffen vermitteln ein Gemeinschaftsgefühl einer Gruppe, auch wenn sie computergestützt über einen Chat abgehalten werden (vgl. [Shotsberger, 2000]).

- Synchrone Methoden werden als zeitlich aufwändig bewertet. Der zeitliche Aufwand nimmt mit der Anzahl der beteiligten Anwender zu (vgl. [Kensing und Blomberg, 2003; Nielsen, 1994a; Sarodnick und Brau, 2006]).

- Mit synchronen Methoden können nur „Momentaufnahmen“ der Situation der Anwender erfasst werden. Sogar bei intensiven Beobachtungen im Nutzungskontext der Anwender (z.B. am Arbeitsplatz) können nur die zum Zeitpunkt beobachteten Ereignisse aufgezeichnet werden (vgl. [Tullis et al., 2002]).

- Der Informationsaustausch erfolgt in Echtzeit, so dass keine Möglichkeiten zum Zurückziehen und Reflektieren besteht. Der Anwender steht unter dem sogenannten „Echtzeitdruck“ (vgl. [Schulmeister, 2006]).

Für die Anwendung der Bewertungskriterien (vgl. Kapitel 2.9) auf synchrone Methoden zur Anwenderbeteiligung existieren in der Literatur eine Fülle an Hinweisen über ihre Effektivität sowie Experten-Meinungen über ihre Stärken und Schwächen (vgl. [Carrizo et al., 2008; Hickey und Davis, 2003; Sarodnick und Brau, 2006]). Allerdings fehlt es an einer vollständig fundierten Evidenz. Ein ausführlicher Vergleich der Methoden unter vergleichbaren Rahmenbedingungen ist in der Literatur nicht auffindbar. Daher kann eine Bewertung lediglich auf theoretischen Überlegungen, Expertenerfahrungen und Evaluationen einzelner Methoden basieren (vgl. [Carrizo et al., 2008; Davis et al., 2006; Dieste und Juristo, 2010; Goguen und Linde, 1993; Hickey und Davis, 2003; Kujala et al., 2005; Nielsen, 1994a; Nielsen, 1994b; Sarodnick und Brau, 2006; Wilson et al., 1996; Wilson et al., 1997; Wixon et al., 2002]). Beim Einsatz von Methoden zur synchronen Anwenderbeteiligung sind darauf aufbauend folgende Auswirkungen zu erwarten (vgl. Tabelle 3.2):

- Der zeitliche Aufwand für die Anwendung von Methoden zur synchronen Anwenderbeteiligung ist im Allgemeinen als „Hoch“ zu bewerten. Für die Vorbereitung sind u.a. die Vorgehensweise zu definieren, die Anwender zu rekrutieren, Kommunikationsmedien zu richten, Termine zu koordinieren, etc. Nach Carrizo [2008] ist hierfür mit einem erhöhten Aufwand zu rechnen. Die Interaktion zwischen Anwender und Entwickler moderiert der Moderator mithilfe von Arbeitstreffen. Zudem sind Abstimmungen mit den Entscheidern notwendig. Da hierfür alle Beteiligten an den Arbeitstreffen teilnehmen müssen, ist auch hier mit einem er-

höhten Aufwand zu rechnen (vgl. [Nielsen, 1994a; Nielsen, 1994b; Wilson et al., 1997]). Die Nachbereitung besteht aus einer schriftliche Fixierung der mündlich formulierten Anwenderbeiträge, der Zusammenfassung der eigenen Notizen zu den Beobachtungen, der Überarbeitung der Kommunikationsmedien, statistische Analysen und die abschließende Auswertung der Ergebnisse. Außerdem sind alle Beteiligten inklusive der Anwender über den Projektverlauf und die getroffenen Entscheidungen zu informieren. auch hierfür ist ein hoher Aufwand zu erwarten (vgl. [Nielsen, 1994a; Sarodnick und Brau, 2006]). Bei Rückfragen und unvollständigen Informationen können zudem weitere Iterationen zur Interaktion mit den Anwendern notwendig werden. Da bei synchronen Methoden keine Rückzugmöglichkeit und reflektierende Betrachtung möglich ist, müssen Wiederholungen bzw. Rückfragen eingeplant werden (vgl. [Nielsen, 1994a]).

- Die Qualität der Anwenderbeiträge bei einer synchronen Anwenderbeteiligung ist im Allgemeinen als gut bzw. „Mittel“ zu bewerten. Es ist zu erwarten, dass die Verständlichkeit der Anwenderbeiträge hoch ist, da sich Anwender und Moderator im direkten Austausch befinden, Unklarheiten zu den Anwenderbeiträgen sofort klären können und der Moderator den Entwicklern die Ergebnisse erläutern kann. In Bezug auf die Konstruktivität der Anwenderbeiträge können Anwender zwar als wertvolle Informationsressource aber weniger als Ideengeber und Lösungsentwickler betrachtet werden (vgl. [Kujala, 2008; Wilson et al., 1997]). Anwender können daher besser auf Fragen bzw. Vorschläge reagieren, als eigene Vorschläge einbringen bzw. radikale Änderungen in Erwägung ziehen (vgl. [Tohidi et al., 2006]). Dies wird durch den „Echtzeitdruck“ der synchronen Kommunikation verstärkt. Mithilfe von Kreativitätstechniken, dem Einsatz von Prototypen und der Diskussion unterschiedlicher Gestaltungsalternativen kann dem entgegengewirkt werden (vgl. [Tohidi et al., 2006]). Daher ist die Konstruktivität der Anwenderbeiträge als „Mittel“ einzuschätzen. Der Umfang der Anwenderbeiträge ist als „Hoch“ einzuschätzen, da der Moderator die fehlenden Informationen nachfragen bzw. aus Beobachtungen selbst ergänzen kann. Die Bewertung der Anzahl der Anwenderbeiträge aufgrund des „Echtzeitdrucks“ beträgt „Mittel“. Die Repräsentativität der Anwenderbeiträge ist „Niedrig“, da mit synchronen Methoden wegen ihrer Ressourcen-Bindung und dem zeitlichen Aufwand lediglich eine begrenzte Anzahl an Anwendern beteiligt werden kann. Der Kontext-Bezug der Anwenderbeiträge ist als „Mittel“ einzustufen, da mit synchronen Methoden lediglich eine „Momentaufnahme“ der Situation der Anwender erfasst werden kann.

- Die Belastung der Anwender ist bei einer synchronen Anwenderbeteiligung im Allgemeinen als „Mittel“ zu bewerten. Die Effizienz der Erstellung von Anwenderbeiträgen ist niedrig, da Anwender zeitlich stark eingebunden werden müssen. Die Komplexität der Erstellung von Anwenderbeiträgen ist hingegen niedrig, da der Anwender seine Beiträge im Regelfall nur mündlich an den Moderator zu formulieren braucht (vgl. [Sarodnick und Brau, 2006]).

Tabelle 3.2 fasst die oben durchgeführte Bewertung von Methoden zur synchronen Anwenderbeteiligung zusammen. In der Gesamtbetrachtung zeigt sich, dass mit Methoden zur synchronen Anwenderbeteiligung hochwertige Anwenderbeiträge erzielt werden können, wobei Verbesserungspotentiale bei Repräsentativität, Konstruktivität, Anzahl und Kontext-Bezug der Anwenderbeiträge liegen. Außerdem ist die Komplexität der Erstellung von Anwenderbeiträgen für den Anwender niedrig, so

dass keine Einstiegsbarrieren existieren sollten. Allerdings ist ein hoher Aufwand für Moderatoren, Entwickler und Anwender zu erwarten.

Parameter	Bewertungskriterien	Synchrone Methoden zur Anwenderbeteiligung
Effizienz der Entwicklung	- Effizienz der Vorbereitung	Niedrig
	- Effizienz der Interaktion mit Anwender	Niedrig
	- Effizienz der Nachbereitung	Niedrig
Qualität der Anwenderbeiträge	- Verständlichkeit der Anwenderbeiträge	Hoch
	- Konstruktivität der Anwenderbeiträge	Mittel
	- Umfang der Anwenderbeiträge	Hoch
	- Anzahl der Anwenderbeiträge	Mittel
	- Repräsentativität der Anwenderbeiträge	Niedrig
	- Kontext-Bezug der Anwenderbeiträge	Mittel
Belastung der Anwender	- Effizienz der Erstellung von Anwenderbeiträgen	Niedrig
	- Komplexität der Erstellung von Anwenderbeiträgen	Niedrig

Tabelle 3.2: Bewertung von Methoden zur synchronen Anwenderbeteiligung

3.3 Fazit zu Kapitel 3

Ziel von Kapitel 3 war es, die Potentiale und Grenzen von Methoden zur synchronen Anwenderbeteiligung zu eruieren. Hierfür wurden ein Überblick zu den existieren Methoden gegeben, Vor- und Nachteile erörtert und darauf aufbauend eine Bewertung anhand der Bewertungskriterien aus Kapitel 2.9 vorgenommen.

Dabei konnten folgende eigene Beiträge herausgearbeitet werden:

- Anhand einer Literaturanalyse wurden Methoden zur synchronen Anwenderbeteiligung identifiziert, die möglichen unterschiedlichen Ausprägungen beschrieben und ihre gemeinsamen Eigenschaften ausgearbeitet. Dabei war es notwendig, aus den Untersuchungen zu den einzelnen Methoden ein Gesamtbild zu entwickeln, das eine Bewertung der Methoden ermöglichte.

- Darauf aufbauend wurden anhand einer Literaturanalyse die Potentiale und Grenzen von Methoden zur synchronen Anwenderbeteiligung gemäß der Bewertungskriterien aus Kapitel 2.9 erfasst. Die Vorteile von Methoden zur synchronen Anwenderbeteiligung liegen in der niedrigen Komplexität der Erstellung von Anwenderbeiträgen für den Anwender sowie die zu erwartende gute Qualität der Anwenderbeiträge. Verbesserungspotentiale finden sich bei Repräsentativität, Konstruktivität, Anzahl und Kontext-Bezug der Anwenderbeiträge. Ihre Grenzen liegen vor allem in ihrer Effizienz der Vor- und Nachbereitung sowie der Interaktion

mit dem Anwender. Zudem ist auch mit einer hohen zeitlichen Belastung der Anwender zu rechnen.

Die Potentiale einer synchronen Anwenderbeteiligung liegen somit in ihrer Effektivität. Nicht ohne Grund stellt das persönliche Interview nach Carrizo et al. [2008] und Dieste et al. [2008; 2010] die effektivste Methode dar. Die niedrige Effizienz der Entwicklung und die hohe Belastung der Anwender können jedoch dazu führen, dass eine Anwenderbeteiligung aufgrund von knappen Ressourcen oder engen Zeitplänen nur in einem begrenzten Umfang umgesetzt wird, obwohl ihre Effektivität nicht in Frage gestellt wird. Zudem sind die Möglichkeiten zur Rekrutierung von Anwendern begrenzt, wenn die Anwender von dem hohen Aufwand abgeschreckt werden und aufgrund des hohen Aufwandes zudem nur eine begrenzte Anzahl an Anwendern eingeladen werden kann.

In diesem Kapitel wurde somit die zweite Forschungsfrage dieser Arbeit nach den Potentialen und Grenzen von Methoden zur synchronen Anwenderbeteiligung beantwortet.

Wie bereits in Kapitel 1.1 motiviert, stellt sich nun die Frage ob Methoden mit einer asynchronen Kommunikation Vorteile gegenüber Methoden mit synchroner Kommunikation besitzen und eine Alternative für die Anwenderbeteiligung in Software-Projekten darstellen. Für diese Arbeit ergibt sich daraus die Notwendigkeit, im nächsten Kapitel die Potentiale und Grenzen von Methoden zur asynchronen Anwenderbeteiligung als Ergänzung zu Methoden zur synchronen Anwenderbeteiligung zu erörtern.

4 Potentiale und Grenzen von Methoden zur asynchronen Anwenderbeteiligung

Ziel dieses Kapitels ist die Untersuchung der Potentiale und Grenzen von Methoden zur asynchronen Anwenderbeteiligung. Hierzu werden zu Beginn ein Überblick über Methoden zur asynchronen Anwenderbeteiligung gegeben und ihre Potentiale und Grenzen aufbauend auf der aktuellen Studienlage und anhand der Bewertungskriterien aus Kapitel 2.9 bewertet.

4.1 Überblick über Methoden zur asynchronen Anwenderbeteiligung

Tabelle 4.1 stellt eine Auflistung von Methoden zur asynchronen Anwenderbeteiligung dar (vgl. [Sarodnick und Brau, 2006]). Hierfür wurden im Rahmen einer Literaturanalyse diejenigen Methoden identifiziert, die eine Vorgehensweise zur Interaktion mit Anwendern beschreiben, in mehreren unterschiedlichen Literaturquellen erwähnt werden und eine asynchrone Kommunikation, also eine zeitlich versetzte Kommunikation zwischen Anwendern und Moderatoren, vorsehen

Die Methode „Culture Probes“ bzw. „Tagebuchstudien“ (vgl. [Dörner et al., 2008; Gaver, 1999]) dient dazu, die Arbeitsumgebung der Anwender zu verstehen und Bedürfnisse und Probleme in ihrer Arbeitsumgebung erfassen. Hierfür protokollieren Anwender den Ablauf ihrer Arbeit und ihre Erfahrungen mit der Software. In web-basierten „Diskussionsforen“ können Anwender über ihre Wünsche und Vorstellungen diskutieren (vgl. [Lewis, 2008]). Sie werden überwiegend in Open-Source-Software-Projekten eingesetzt. Mit web-basierten Wikis können Anwender über Hypertext-Dokumente ihr Wissen einbringen, ihre Anforderungen beschreiben und an Diskussionen teilnehmen (vgl. [Decker et al., 2007]). Mithilfe eines Helpdesks können Anwender per Email, Telefon oder Issue-Tracker Probleme melden (vgl. [Prause et al., 2008]). Mit einem Fragebogen kann der Moderator gezielt nach bestimmten Komponenten bzw. Themen fragen (vgl. [Sarodnick und Brau, 2006]). Eine Variation des Usability Test stellt der „Remote Asynchronous Usability Test“ dar, bei der die Anwender über ein verteiltes Kommunikationssystem (im Regelfall über das Internet) an einem Usability Test teilnehmen (vgl. [Hartson et al., 1996]). Sie testen zu einem beliebigen Zeitpunkt die Software und füllen anschließend einen webbasierten Fragebogen aus. Außerdem können sie jederzeit über eine Schaltfläche auf dem Anwendungsbildschirm eine Nachricht zur aktuellen Situation senden, den sogenannten „critical incidents“. Zudem kann mithilfe einer Aufzeichnungssoftware das Verhalten des Anwenders, z.B. durch eine Aufnahme des Bildschirms, Mausbewegungen und Tastatureingaben sowie Videoaufnahme der Anwender, protokolliert werden. Bei einem „automatisierten Problembericht“ besitzt die Software hingegen eine interne Fehlerberichterstattung zur internen Qualitätssicherung. Ereignet sich während der Nutzung ein Fehler bzw. ein Problem, wird ein Fehlerbericht erstellt

und mit Erlaubnis des Anwenders an den Hersteller der Software übermittelt (vgl. [Dray und Siegel, 2004; Hilbert und Redmiles, 1999]).

Methode	Kurzbeschreibung
Remote Asynchronous Usability Test	Über ein räumlich und zeitlich verteiltes Kommunikationssystem kann ein Usability Test mit Anwendern durchgeführt werden.
Automatisierter Problembericht	Die Software sendet eigenständig bei Fehlermeldungen bzw. Ausnahmebehandlungen („Exceptionhandling“) einen Bericht an den Entwickler.
Fragebogen	Anwender geben schriftlich ihre Meinung zu definierten Fragestellungen ab.
Culture Probes/ Tagebuchstudien	Anwender führen über ihren Arbeitsalltag Tagebuch mithilfe von Notizblöcken, Fotokamera, Aufnahmegerät, etc.
Diskussionsforen	In webbasierten Diskussionsforen tauschen sich Anwender über Themen aus und diskutieren Verbesserungsmöglichkeiten für Software.
Wiki	Über ein webbasiertes Wiki-System können Anwender ihre Beiträge in Form von Hypertext-Dokumenten einbringen.
Helpdesk	Mit webbasierten Issue-Trackern, Email und/oder Telefon können Anwender Wünsche und Verbesserungsvorschläge melden.

Tabelle 4.1: Methoden zur asynchronen Anwenderbeteiligung in Software-Projekten

Allen Methoden zur asynchronen Anwenderbeteiligung ist gemeinsam, dass die Anwender ihre Beiträge zu einem selbst gewählten Zeitpunkt zeitlich versetzt aufzeichnen bzw. schriftlich formulieren und den Moderatoren zur Verfügung stellen. Beim automatisierten Problembericht und der Marktanalyse kann eine Anwenderbeteiligung sogar ohne eine aktive Teilnahme der Anwender erfolgen. Die Methoden unterscheiden sich u.a. in der Form, wie Anwender ihre Beiträge verfassen, im Freiheitsgrad zur Formulierung ihrer Beiträge, in dem Unterstützungsgrad, mit dem Anwendern bei der Formulierung ihrer Beiträge assistiert wird, und ob auch Diskussionen zwischen Anwendern untereinander ermöglicht werden.

4.2 Bewertung von Methoden zur asynchronen Anwenderbeteiligung

Aus den Erfahrungen mit asynchroner Kommunikation in der computergestützten Gruppenarbeit (vgl. [Hasenkamp et al., 1997; Schwabe et al., 2001]), auch bekannt als Computer-Supported Cooperative Work (CSCW), im E-Learning bzw. im computergestützten Gruppenlernen (vgl. [Haake et al., 2004]), dem sog. Computer-Supported Cooperative Learning (CSCL), und im Remote Usability Test (vgl. [Andreasen et al., 2007]) lassen sich folgende Eigenschaften einer computergestützten asynchronen Kommunikation ableiten:

- Mit einer asynchronen Kommunikation ist immer eine Speicherung der Kommunikation verbunden. Dies erfolgt entweder schriftlich über Text bzw. Abbildungen oder audiovisuell über Aufzeichnungen. Somit ist eine asynchrone Kommunikation stets persistent bzw. permanent (vgl. [Schwabe et al., 2001]).

- Als elektronische asynchrone Kommunikationsmedien werden nach Schwabe et al. [2001] üblicherweise E-Mail, Diskussionsforen, Wikis und Dokumentenablagen verwendet, die alle vorwiegend auf einer schriftlichen Kommunikation basieren. Alternativ können audiovisuelle Medien wie z.B. Audio- und Videoaufzeichnungen produziert werden.

- Die Kommunikanten sind „*vom Druck der Echtzeit befreit*“ (vgl. [Schulmeister, 2006, S. 162]). Die asynchrone Kommunikation erfolgt sorgfältiger (vgl. [Meyer, 2003]) und formaler (vgl. [Lapadat, 2002]) als bei einer synchronen Kommunikation, da mehr Zeit zum Denken und Formulieren vorhanden ist. Nachrichten können „*langsamer produziert und langsamer aufgenommen*“ (vgl. [Schulmeister, 2006, S. 163]) werden. Somit eignet sich eine asynchrone Kommunikation auch für kognitiv anspruchsvolle Tätigkeiten und kritisches Denken, sogenannte Denkprozesse höherer Ordnung (vgl. [Lapadat, 2002]). Nach Mejias [2005] können auch komplexe Strukturen mit der asynchronen Kommunikation aufgebaut werden, um Sachverhalte ausführlich und strukturiert darzustellen.

- Dank der Persistenz können Nachrichten mehrfach eingesehen und nachträglich reflektiert werden (vgl. [Garrison, 2003]). Zudem können Verknüpfungen, sog. Hyperlinks, zwischen den gespeicherten Inhalten hergestellt werden, um auf andere Beiträge referenzieren zu können (vgl. [Mejias, 2005]). Dadurch findet automatisch auch eine Archivierung der Kommunikation statt, die für Außenstehende die Möglichkeit bietet, die erfolgte Kommunikation auch nachträglich einsehen zu können (vgl. [Mazur, 2004]). Dies steigert die Möglichkeiten zur Partizipation für andere, die zu einem späteren Zeitpunkt hinzukommen (vgl. [Hew und Cheung, 2003]).

- Soziale Barrieren können bei einer asynchronen Kommunikation verringert sein, da der Sender der Nachricht zeitlich versetzt und somit seine Nachricht im Regelfall mit mehr Zeit und ohne Anwesenheit des Empfängers formulieren kann (vgl. [Hew und Cheung, 2003]).

- Eine Antwort bzw. Feedback kann in einer asynchronen Kommunikation auch stark zeitlich versetzt, unter Umständen sogar mehrere Tage später erfolgen. Dies hat den Nachteil, dass eine Diskussion bzw. der Austausch von Informationen über eine längere Zeit verlaufen kann und Gesprächsthemen möglicherweise erst sehr spät geklärt werden (vgl. [Hew und Cheung, 2003]). Andererseits kann dies auch ein Vorteil sein, wenn eine sofortige Reaktion nicht erwünscht ist und eine weitere Kommunikation vermieden werden soll: „Der asynchrone Austausch erleichtert nicht nur die Kommunikation, sondern auch die Kommunikationsvermeidung“ (vgl. [Döring, 2003, S. 52]).

- Eine computergestützte schriftliche Kommunikation reduziert die Kanäle einer Kommunikation und kann zu Missverständnissen führen. Sie „*unterliegt dem Phänomen der Kanalreduzierung, d.h. dem Ausfall paralinguistischer Eigenschaften der Kommunikation (Lautstärke, Stimmhöhe, Sprechtempo, Artikulation, Klangfarbe, etc.), nonverbaler Botschaften (Körper-*

und Kopfbewegung, Mimik, Gestik, etc.) sowie extralinguistischer Signale (Emotionen, Sprechereigenschaften, etc.)“ (vgl. [Schulmeister, 2006, 148]).

Aus diesen Eigenschaften lassen sich mehrere mögliche Vorteile einer asynchronen Kommunikation in der Anwenderbeteiligung ableiten:

- Anwender können bei einer asynchronen Kommunikation ihre Meinung dann einbringen, wenn ihnen etwas auffällt bzw. einfällt. So können sich Anwender in aller Ruhe bzw. zu einem passenden Zeitpunkt Gedanken machen. Dabei können sie in ihrem realen Nutzungskontext bleiben und auf Probleme bzw. Lösungen in ihrem Arbeitsprozess achten (vgl. [Andreasen et al., 2007; Hartson et al., 1996; Hartson und Castillo, 1998]).

- Die Teilnahme der Anwender kann mit einer asynchronen Kommunikation vereinfacht werden, da keine festen Termine vorausgesetzt werden. Die Anwender können somit orts- und zeitunabhängig teilnehmen. Zum einen können so mehr Anwender beteiligt werden, zum anderen können Anwender ihre Teilnahmezeiten an ihrer terminlichen Auslastung ausrichten und sich mehr Zeit für ihre Projektbeteiligung nehmen (vgl. [Andreasen et al., 2007; Burger et al., 2008; Hartson et al., 1996]).).

- Bei einer asynchronen Kommunikation überträgt der Anwender seine Nachrichten entweder als Sprach- bzw. Videoaufzeichnung oder als schriftliches Dokument. Für den Entwickler gibt sich daraus der Vorteil, dass er den Beitrag des Anwenders in einer persistenten Form erhält. Während des Testens eines Prototypen bzw. einer Software können zudem Anwendungsdaten wie z.B. die Protokollierung des Benutzungsverhaltens des Anwenders aufgezeichnet und übertragen werden (vgl. [Andreasen et al., 2007]).

- Anwender können auch nach Einführung einer Software und somit nach Abschluss eines Software-Projektes Verbesserungsvorschläge und Fehler melden, und somit einen kontinuierlichen Verbesserungsprozess ohne aktive Betreuung der Anwender initiieren, bei der abhängig vom Bedarf der Anwender nachträgliche Anpassungen erfolgen können (vgl. [Hartson et al., 1996; Hartson und Castillo, 1998]).

Mit der Anwendung von asynchroner Kommunikation in Software-Projekten sind jedoch auch mehrere Probleme verbunden:

- Die Aufmerksamkeit der Anwender ist bei einer asynchronen Kommunikation schwieriger zu gewinnen als bei einer synchronen Kommunikation (vgl. [Andreasen et al., 2007]). Wenn die Anwender sich den Zeitpunkt ihrer Beteiligung selbst aussuchen dürfen, besteht die Gefahr, dass sie andere Aktivitäten für wichtiger befinden und schlimmstenfalls keine Gelegenheit zur Beteiligung finden. Bei einer synchronen Kommunikation ist ein zeitlicher Rahmen festgelegt, innerhalb dem die volle Aufmerksamkeit des Anwenders sichergestellt werden kann.

- Bei einer asynchronen Kommunikation erfolgt die Kommunikation im Regelfall langsam und zeitlich verzögert, da die Nachrichten erst schriftlich formuliert werden müssen. Für den Anwender bedeutet dies, dass er seine Vorstellungen exakt formulieren muss, um seine Ideen und Meinungen einbringen zu können. Im gleichen Zug kann der Moderator nicht in Echtzeit auf Besonderheiten reagieren, so dass hier der Anwender auf sich allein gestellt ist.

- Das Auftreten von Missverständnissen ist ein bekanntes Phänomen bei einer asynchronen Kommunikation, da zum einen die Kommunikationskanäle stark reduziert sind und zum anderen der Aufwand zur schriftlichen Formulierung so hoch ist, dass nicht alles im Detail beschrieben wird (vgl. [Andreasen et al., 2007]). Für die Effizienz und Effektivität der Kommunikation spielt dieser Umstand eine zentrale Rolle, da sich Diskussionen bei einer asynchronen Kommunikation aufgrund von zeitlichen Verzögerungen als aufwändig erweisen können. Vor allem für die Anwender ist eine asynchrone Anwenderbeteiligung mit einem erhöhten Zusatzaufwand verbunden, da sie ihre Beiträge selbstständig ohne Unterstützung eines Moderators verfassen und einbringen müssen. Außerdem können die Emotionen und subjektiven Reaktionen der Anwender in einer asynchronen Anwenderbeteiligung nur in einem geringen Maße erfasst werden. Die Folge ist, dass hinter den Beiträgen der Anwender kein Eindruck bzw. kein „Gesicht" steht und damit eine Einschätzung der Situation und der Priorität des Beitrags erschwert wird (vgl. [Dray und Siegel, 2004]).

- Interkulturelle bzw. internationale Software-Projekte stellen nach Dray und Siegel [2004] eine besondere Herausforderung für eine asynchrone Anwenderbeteiligung dar. Die Kommunikation ist durch kulturelle und sprachliche Unterschiede bereits herausfordernd und ein intensiver Austausch unerlässlich. Durch eine computergestützte Kommunikation wird eine weitere Ebene hinzugefügt, die zu Missverständnissen führen kann (vgl. [Dray und Siegel, 2004; Dray und Karat, 1994]). Eine asynchrone Anwenderbeteiligung sollte daher nach Dray und Siegel [2004] auch nicht für formative Evaluationen angewendet werden. Nach ihren Erfahrungen sind persönliche Gespräche für qualitative Erhebungen effektiver.

Für die Anwendung der Bewertungskriterien (vgl. Kapitel 2.9) auf asynchrone Methoden zur Anwenderbeteiligung existieren in der Literatur nur wenige Anhaltspunkte (vgl. [Andreasen et al., 2007; Bruun et al., 2009]). Daher basiert die Bewertung auf den oben beschriebenen Vor- und Nachteilen sowie auf Experimenten zum Remote Usability Test (vgl. [Andreasen et al., 2007; Andrzejczak und Liu, 2010; Baker et al., 2007; Bruun et al., 2009; Dray und Siegel, 2004; Hartson et al., 1996; Hartson und Castillo, 1998; Thompson et al., 2004; Tullis et al., 2002]). Folgende Auswirkungen sind demnach mit Methoden zur asynchronen Anwenderbeteiligung zu erwarten (vgl. Tabelle 4.2):

- Der zeitliche Aufwand für die Anwendung von Methoden zur asynchronen Anwenderbeteiligung soll nach Bruun et al. [2009] im Vergleich zu synchronen Methoden niedriger sein. Während sich die Vorbereitungszeit nur wenig verringert, finden sich deutliche Zeiteinsparungen bei der Durchführung und Nachbereitung, da sich viele Arbeitsschritte erübrigen. Dies ist u.a. darauf zurückzuführen, dass die Transkription von Interviews und Fragebögen, die Auswertung von Videoaufzeichnungen und die Anwesenheit während des Testes für den Moderator entfallen.

- Basierend auf den Untersuchungen von Bruun et al. [2009] und Andreasen et al. [2007] ist die Qualität der Anwenderbeiträge bei einer asynchronen Anwenderbeteiligung im Allgemeinen als „Mittel" zu bewerten. Die Verständlichkeit der Anwenderbeiträge verhält sich ähnlich zu synchronen Methoden. Die Konstruktivität und Anzahl der Anwenderbeiträge stellten sich als „Niedrig" heraus. Die fehlende Konstruktivität wurde dadurch sichtbar, dass die meisten asynchron eingebrachten Anwenderbeiträge kosmetische Änderungen an der Benutzeroberfläche der getesteten Software betrafen und sich die wenigsten Beiträge auf kritische

Anforderungen, den sogenannten „critical incidents“, bezogen. Die geringe Anzahl der Beiträge wird darin begründet, dass ohne die „thinking aloud“ Technik eine Vielzahl an Problemen mit der Software nicht gemeldet wird. Beim Umfang der Anwenderbeiträge wurden bei synchronen und asynchronen Methoden vergleichbare Ergebnisse („Mittel“) festgestellt. Die Repräsentativität der Anwenderbeiträge ist als „Hoch“ zu bewerten, da mit asynchronen Methoden eine größere Anzahl von Teilnehmern einbezogen werden kann. Zudem können die Anwender sich auch untereinander austauschen und ihre Beiträge gegenseitig zur Diskussion stellen. Der Kontext-Bezug der Anwenderbeiträge ist „Hoch“, da sich Anwender mit asynchronen Methoden im Nutzungskontext und über einen längeren Zeitraum beteiligen können.

- Die Belastung der Anwender ist bei einer asynchronen Anwenderbeteiligung als „Mittel“ zu bewerten. Für die Erstellung ihrer Anwenderbeiträge können sich die Anwender selbst den Zeitpunkt der Beteiligung wählen und sparen sich zudem Wartezeiten und Anreisewege. Im Vergleich zu synchronen Methoden ist die Effizienz höher und daher als „Mittel“ einzustufen. Die Komplexität der Erstellung von Anwenderbeiträgen ist „Mittel“, da die Anwender ihre Beiträge im Regelfall schriftlich einbringen und sich länger überlegen müssen, wie sie ihre Meinung einbringen. Außerdem ist bei asynchronen Methoden der Einsatz von computergestützten Kommunikationssystemen notwendig, die eine Einarbeitung erforderlich machen (vgl. [Andreasen et al., 2007]).

Parameter	**Bewertungskriterien**	**Asynchrone Methoden zur Anwenderbeteiligung**
Effizienz der Entwicklung	- Effizienz der Vorbereitung	Niedrig
	- Effizienz der Interaktion mit Anwender	Hoch
	- Effizienz der Nachbereitung	Hoch
Qualität der Anwenderbeiträge	- Verständlichkeit der Anwenderbeiträge	Mittel
	- Konstruktivität der Anwenderbeiträge	Niedrig
	- Umfang der Anwenderbeiträge	Mittel
	- Anzahl der Anwenderbeiträge	Niedrig
	- Repräsentativität der Anwenderbeiträge	Hoch
	- Kontext-Bezug der Anwenderbeiträge	Hoch
Belastung der Anwender	- Effizienz der Erstellung von Anwenderbeiträgen	Mittel
	- Komplexität der Erstellung von Anwenderbeiträgen	Mittel

Tabelle 4.2: Bewertung von Methoden zur asynchronen Anwenderbeteiligung

Tabelle 4.2 fasst die oben durchgeführte Bewertung von Methoden zur asynchronen Anwenderbeteiligung zusammen. In der Gesamtbetrachtung zeigt sich, dass mit Methoden zur asynchronen Anwenderbeteiligung hochwertige Anwenderbeiträge erzielt werden können, wobei Verbesserungspotentiale bei Verständlichkeit, Konstruktivität, Umfang und Anzahl der Anwenderbeiträge liegen. Außerdem ist die Komplexität der Erstellung von Anwenderbeiträgen für den Anwender niedrig, so dass

keine Einstiegsbarrieren existieren sollten. Allerdings ist ein hoher Aufwand für Moderatoren, Entwickler und Anwender zu erwarten.

4.3 Fazit zu Kapitel 4

Ziel von Kapitel 4 war es, die Potentiale und Grenzen von Methoden zur asynchronen Anwenderbeteiligung zu erörtern. Hierfür wurden ein Überblick zu den existieren Methoden gegeben, Vor- und Nachteile erörtert und darauf aufbauend eine Bewertung anhand der Bewertungskriterien aus Kapitel 2.9 vorgenommen.

Dabei konnten folgende eigene Beiträge herausgearbeitet werden:

- Anhand einer Literaturanalyse wurden Methoden zur asynchronen Anwenderbeteiligung identifiziert, die möglichen unterschiedlichen Ausprägungen beschrieben und ihre gemeinsamen Eigenschaften ausgearbeitet.
- Anhand einer Literaturanalyse wurden die Potentiale und Grenzen von Methoden zur asynchronen Anwenderbeteiligung gemäß den Bewertungskriterien aus Kapitel 2.9 diskutiert. Die Vorteile von Methoden zur asynchronen Anwenderbeteiligung liegen in der hohen Effizienz bei der Interaktion mit dem Anwender und der anschließenden Nachbereitung sowie der Repräsentativität sowie dem Kontext-Bezug der Anwenderbeiträge. Allerdings ist mit einer im Vergleich zu synchronen Methoden geringen Verständlichkeit, Anzahl und Konstruktivität der Anwenderbeiträge und einer höheren Belastung der Anwender zu rechnen. Die Effizienz der Vorbereitung unterscheidet sich nicht von Methoden zur synchronen Anwenderbeteiligung und ist somit niedrig.

Die Potentiale einer asynchronen Anwenderbeteiligung liegen in ihrer Effizienz und der Integration in den Arbeitsalltag. Daher können mehr Anwender beteiligt und mehr Informationen in ihrem Nutzungskontext erfasst werden. Allerdings ist mit einer geringen Verständlichkeit und Anzahl der Anwenderbeiträge zu rechnen, da die Anwender bei der Erstellung von Anwenderbeiträgen auf sich gestellt sind und der Moderator nicht aktiv eingreifen kann. Zudem kommt eine höhere Belastung auf die Anwender zu, da sie ihre Beiträge schriftlich formulieren müssen.

Vergleichbare Erkenntnisse finden sich bei den Untersuchungen von Lloyd et al. [2002]. Sie kamen zu der Schlussfolgerung, dass eine asynchrone Kommunikation die Effektivität der Anforderungserhebung verringert und synchrone Methoden wie persönliche Gespräche und Besprechungen vorteilhafter sind. Auch Andreasen et al. [2007] und Bruun et al. [2009] mussten feststellen, dass mit asynchronen Methoden trotz einer höheren Anzahl von Anwendern und einer längeren Laufzeit keine gleichwertigen Ergebnisse wie bei synchronen Methoden erzielt werden können. Dennoch stellen asynchrone Methoden ihrer Einschätzung nach eine effizientere Alternative zu synchronen Methoden dar. Sie eignen sich auch als eine sinnvolle Ergänzung zu synchronen Methoden, um unterschiedliche Arten von Anwenderbeiträgen erfassen zu können. Allerdings müssen entweder eine geringe Effektivität in Kauf genommen oder weitere Untersuchungen durchgeführt und neue Ansätze entwickelt werden.

Für die Entwicklung neuer Ansätze sollte bei einer asynchronen Anwenderbeteiligung somit primär die Unterstützung der Anwender bei der Erstellung von Beiträgen im Vordergrund stehen, um deren Verständlichkeit, Anzahl und Konstruktivität zu erhöhen sowie die Belastung der Anwender zu reduzieren. Dabei gilt es zu verhindern, dass die hohe Effizienz der Entwicklung in Mitleidenschaft gezogen wird.

Die bisherigen Methoden setzen vorwiegend auf eine textbasierte Kommunikation. Dies kann eine mögliche Schwachstelle darstellen und somit eine Ansatzstelle für Verbesserungsmöglichkeiten sein. Die Formulierung von Texten zur Beschreibung ihrer Wünsche und Vorstellungen erfordert von den Anwendern die Fähigkeit, das Vokabular zu Software bzw. technischen Systemen zu beherrschen und eine Übersicht über die technischen Möglichkeiten zu haben, um damit die vorliegende Situation beschreiben und ihre Anforderungen präzise und vollständig ausdrücken zu können. Der Einsatz von Annotationssystemen stellt hierfür möglicherweise einen neuen, erfolgversprechenden Ansatz dar. Neben textlichen Formulierungen kommen dabei grafische Ausdruckmittel zum Einsatz, mit deren Hilfe den Anwendern die Erstellung ihrer Beiträge erleichtert werden soll. Mithilfe eines Annotationssystems fügen die Anwender Annotationen auf der grafischen Benutzeroberfläche der getesteten Software ein. Die Anwender sollen sich damit die Formulierung langer Texte ersparen und mit einfachen grafischen Ausdrucksmitteln schnell und präzise auf den Punkt kommen können.

In den Arbeiten von Moore und Shipman [2000; 2001], Tohidi et al. [2006], Naghsh und Dearden [2004] sowie Stevens und Draxler [2006] finden sich grundlegende Hinweise über mögliche Potentiale von Annotationssystemen, die im Folgenden kurz erläutert werden.

Bei Moore und Shipman [2000; 2001] konnten Anwender mithilfe eines grafischen Editors eigene grafische Benutzeroberflächen zusammenstellen und Eigenschaften von Schaltflächen und Dialogen textlich beschreiben. Im Vergleich zu einem textbasierten Fragebogen erstellten die Studienteilnehmer mit dem grafischen Editor eine höhere Anzahl von Anwenderbeiträgen. Die Konstruktivität war ebenfalls höher. Allerdings verbrachten die Studienteilnehmer auch mehr Zeit mit dem grafischen Editor. Ihre Schlussfolgerung lautet (vgl. [Moore und Shipman, 2001, S. 8]):

> *„In many cases the information relied on both graphical and textual aspects of the interface constructs. This indicates that combining the 'language of the GUI' and textual argumentation leads to information that could not have been elicited with either text or graphics alone."*

Die Anwenderbeiträge beziehen sich sowohl auf grafische als auch textliche Aspekte der Benutzeroberfläche einer Software. Die Kombination der „Sprache der grafischen Benutzeroberfläche" mit textlichen Formulierungen führt zu Informationen, die weder mit Text oder Grafiken alleine erfasst werden können.

Tohidi et al. [2006] untersuchten die Effekte von Zeichnungen („user sketches"), welche von Anwendern zur Veranschaulichung ihrer Ideen erstellt werden. Sie stellten fest, dass diese Technik die Anwender zur Reflektion ihrer Anforderungen und Bedürfnisse motiviert und zu einer Verbesserung der Erstellung und Kommunikation von Gestaltungsideen führen kann. Daraus ergeben sich konstruktivere und umfangreichere Anwenderbeiträge. Ähnliche Ergebnisse finden sich auch bei Naghsh und Dearden [2004; 2005]. Bei ihren Studien konnten Anwender Kommentare auf die Benutzeroberfläche

von elektronischen Papier-Prototypen einfügen und über die grafische Benutzeroberfläche Diskussionen mit anderen Anwendern und den Moderatoren führen.

Stevens und Draxler [2006] befassten sich mit einer Werkzeugunterstützung zur Partizipation im Nutzungskontext und entwickelten das sog. „PaDU Plugin“. Um einen Verbesserungsvorschlag zu einer getesteten Software zu melden, können Anwender über eine Schaltfläche in der Software das PaDU Plugin starten und einen Bericht in Form eines Formulars senden. Ihre Evaluation des PaDU Plugin führte u.a. zu der Schlussfolgerung, dass grafische Ausdrucksmittel wie Annotationen auf Bildschirmfotos hilfreich sein können (vgl. [Stevens und Draxler, 2006, S. 91]):

> *„Auf Grund unserer Erfahrung mit der Benutzung des PaDU Plugin gilt es, Nutzern nicht nur textliche Ausdrucksmittel, sondern insbesondere auch graphische Ausdrucksmittel zur Verfügung zu stellen. Mittels einer Nutzungskamera soll der Benutzer in die Lage versetzt werden ein Bildschirmfoto des aktuellen Systems anzufertigen und mittels eingebauten Graphikwerkzeugs umzugestalten und zu annotieren.“*

Aus diesen Überlegungen heraus stellt sich die Frage, ob Annotationssysteme die festgestellten Nachteile von Methoden zur asynchronen Anwenderbeteiligung kompensieren können. Sollte dies der Fall sein, bietet sich die Entwicklung einer neuen Methode zur asynchronen Anwenderbeteiligung an, bei der Annotationssysteme zur Kommunikation zwischen Anwendern und Moderatoren herangezogen werden sollten. Daher werden im nächsten Kapitel 5 die Potentiale von Annotationssystemen in der asynchronen Anwenderbeteiligung untersucht und es wird der Frage nach der Notwendigkeit der Entwicklung einer neuen Methode zur asynchronen Anwenderbeteiligung nachgegangen.

5 Potentiale von Annotationssystemen zur asynchronen Anwenderbeteiligung

In diesem Kapitel werden die Begrifflichkeiten zu Annotationssystemen und die unterschiedlichen Einsatzmöglichkeiten von Annotationen beschrieben. Darüber hinaus werden das breite Spektrum an Annotationssystemen vorgestellt und die bisherigen Ansätze zur Unterstützung der Anwenderbeteiligung in Software-Projekten erläutert. Anschließend erfolgt eine Diskussion der Vor- und Nachteile von Annotationssystemen zur asynchronen Anwenderbeteiligung und es wird der Frage nach der Notwendigkeit der Entwicklung einer neuen Methode zur asynchronen Anwenderbeteiligung nachgegangen.

5.1 Annotationen und Annotationssysteme

Der Begriff des Annotierens ist auf das Lateinische „annotare“ bzw. „annotatio“ zurückzuführen und bedeutet „Aufzeichnen“ bzw. „Aufzeichnung, Vermerk“. Nach der Brockhaus Enzyklopädie wird eine Annotation als *„erläuternde Anmerkung, Aufzeichnung; kurze Charakterisierung eines Buches für bibliothekar. Zwecke“ verstanden*. Die Tätigkeit des Annotierens liegt demnach vor, wenn ein Objekt mit einem Vermerk bzw. einer Annotation ergänzt wird. Eine Annotation ist dabei stets auf eine bestimmte Stelle eines Dokumentes bezogen - auf ein Wort, eine Wortgruppe, einen Satz, einen ganzen Abschnitt, eine Abbildung oder einen Teil einer Abbildung (vgl. [Ovsiannikov et al., 1999; Schilit et al., 1998]). Eine Annotation besteht somit aus einem Bezugspunkt („anchor“) und einem Inhalt („content“) [Brush et al., 2001]. Die Position der Annotation kann sowohl direkt im Dokument als auch separat außerhalb des Dokuments erfolgen. Die annotierte Stelle muss auf jeden Fall einen Verweis zu ihrer Annotation enthalten. Bei Papierdokumenten kann beispielsweise ein Zettel vor die betreffende Seite mit einem entsprechenden Hinweis gelegt oder an eine bestimmte Stelle eines Dokuments geheftet werden. Bei elektronischen Dokumenten kann die Annotation direkt im Dokument oder in einem separaten Annotationsfenster angezeigt werden. Als Attribute einer Annotation werden in der Literatur die Merkmale Klasse („class“), Typ („typ“), Titel („title“), Kontext („context“), Nummer („id“), Zeitpunkt („time“), Autor („annotator“), Status („status“) und Priorität („priority“) genannt (vgl. [Catlin et al., 1989; Kahan und Koivunen, 2001; Margolis und Resnick, 1999; Weng und Gennari, 2004; Zheng et al., 2006]).

Bei der Art und Weise der Annotation lassen sich nach Marshall [1998] die vier Arten „Textauswahl“ (z.B. farblich abgehoben), „symbolhafte Hervorhebung“ (z.B. Ausrufezeichen), „satzförmige Kommentare“ sowie „bildhafte Darstellungen“ (z.B. Zeichnungen) unterscheiden. Darüber hinaus kann ein Vermerk bzw. eine Annotation u.a. auch als Sprach- oder Videoaufzeichnung auftreten. Charakteristisch für Annotationen ist somit, dass sie in das annotierte Dokument eingreifen und dabei nur im Kontext ihres Dokumentes eine Funktion besitzen. Ohne den Kontext ist eine Annotation funktionslos.

Nach Marshall [Marshall, 1998; 2000] können Annotationen mit Hilfe von sieben Beschreibungsdimensionen charakterisiert werden (vgl. Tabelle 5.1). Die Form einer Annotation kann in ihrer Formalität (formal vs. informal) und in ihrer Explizitheit (explizit vs. implizit) variieren. Eine formale Annotation entspricht einer bestimmten Formatvorgabe, für eine informale Annotation gibt es dagegen keinerlei Beschränkungen. Informale Annotationen geben dem Leser mehr Freiheit, sind jedoch maschinell nur schwer zu verarbeiten. Die Explizitheit einer Annotation gibt an, wie verständlich sie für andere Personen außer dem Autor bzw. Leser/Schreiber ist. Da der Autor eine Annotation nicht expliziter anfertigt, als im Moment des Lesens unbedingt erforderlich ist, sind persönliche Annotationen in der Regel unvollständig, für andere Personen häufig nicht verständlich und somit implizit. Oft werden Gedanken nur angedeutet oder individuelle Kürzel verwendet. Um zu gewährleisten, dass auch andere Personen eine Annotation verstehen, muss sie ausführlicher sein. Neben der Explizitheit hilft anderen Lesern auch ein höherer Formalisierungsgrad beim Verstehen.

Kategorie	**Dimension**	**Ausprägung**
Form	Formalität	Formal vs. Informal
	Explizitheit	Explizit vs. Implizit
Funktion		Schreiben vs. Lesen
		Extensiv vs. Intensiv
	Permanenz	Permanent vs. Flüchtig
Kommunikation	Adressat	Öffentlich vs. Privat
	Empfängerkreis	Global vs. Institutionell vs. Arbeitsgruppe vs. Persönlich

Tabelle 5.1: Beschreibungsdimensionen von Annotationen [Marshall, 2000]

Die Beschreibungsdimension „Schreiben vs. Lesen“ beschreibt, inwieweit Annotationen selbst ein Stück Text sind. Da Annotationen gleichzeitig ein integraler Bestandteil des Lesens als auch ein Akt des Schreibens sind, transformieren sie den Leseprozess in einen gemischten Lese-Schreib-Prozess. Sowohl Lesen als auch Schreiben sind konstruktive Prozesse, bei denen durch die Interaktion mit dem Text Bedeutung geschaffen wird (vgl. [Tierney und Pearson, 1983]). Annotationen, die einen Text optisch gliedern, z.B. Unterstreichungen oder Gliederungszahlen am Rand, sind eher als Leseprozess zu verstehen, da sie diesen unterstützen. Kommentare und Meinungsäußerungen, wie Fragezeichen, Verweise oder kurze Sätze, hingegen sind eher als schreibend zu verstehen. Die Extensivität einer Annotation sagt aus, ob sie über verschiedene Texte hinweg geht (zum Beispiel Querverweise) oder vertiefend innerhalb eines Textes verbleibt. Flüchtige Annotationen schließlich sind nur für den Moment gedacht, quasi als Lesehilfe. Permanente Annotationen sollen wiederholt gelesen werden oder für andere zur Verfügung stehen. Annotationen können darüber hinaus hinsichtlich ihres Adressats unterschieden werden und entweder für die Öffentlichkeit (z.B. kommentierte Gesetztestexte) oder für private Zwecke bestimmt sein. Marshall differenziert die intendierten Empfänger einer Annotation noch weiter zwischen „global“, „institutionell“, „Arbeitsgruppe“ und „persönlich“.

Der Begriff der Annotation wird allerdings in der Literatur unterschiedlich verwendet und häufig synonym mit Notizen oder Markierungen gehandhabt. Fish et al. [1988] definieren eine Annotation als Hypertextknoten, der zu seinem Basisdokument verknüpft ist. In der Publikation von Ovsiannikov et al. werden Annotation als „Klumpen" („clumps") bzw. Gruppierung von Kommentaren verstanden, die sich auf mehrere Orte in einem Dokument beziehen. Schilit et al. [1998, S. 251] verwenden den Begriff Markierung bzw. Aufzeichnung („marks") und unterteilen eine Aufzeichnung in Annotationen und Notizen:

> *„We use annotation to refer to marks on (or attached to) reading matter, and notes to mean marks that are not collocated with the text to which they refer."*

Annotationen beziehen sich demnach auf den Text, während Notizen nicht mit dem Text in Verbindung stehen müssen. Eine Abgrenzung zum Begriff Notiz und Aufzeichnung führt auch Riedewald [2003] durch. Sie betont, dass die Unterscheidung zwischen Notiz und Annotation notwendig ist. Aus ihrer Sicht ist es das Ziel einer Notiz *„das Lesen im Text überflüssig zu machen"* (vgl. [Riedewald, 2003, S. 13]). Eine Annotation hingegen funktioniert nur mit einem konkreten Dokument und ist in das Dokument integriert. Riedewald [2003, S. 13] unterscheidet Annotationen und Notiz schließlich folgendermaßen:

> *„Während Annotationen Textergänzungen bzw. Textumformungen sind, sind Notizen neu kombinierbare, selbständige ‚Wissensstücke'. Annotationen sind eine Lesehilfe, ein erster Schritt und Vorbereitung auf eine weitere, tiefer gehende Verarbeitung. Sie erfüllen eine sehr grundlegende Funktion beim Umgang mit Dokumenten: Auf ihrer Grundlage entstehen später die Notizen. Notieren ist nicht in den Leseprozess integriert. Die frei kombinierbaren Wissensstücke, wie sie Notizen darstellen, oft auf Karteikarten zusammengestellt, lassen sich dagegen gut sortieren und neu strukturieren und bieten somit ideale Voraussetzungen für das Lernen, Erstellen von Texten und Referaten sowie für die Informationssuche."*

Somit ist eine Annotation nach Riedewald [2003] eingewoben in den Kontext und inhaltlich und bezüglich ihrer Lokalisierung abhängig von dem konkret annotierten Teil der Repräsentation. Sie stellt einen Zusatz zu einer Repräsentation dar. Eine Annotation kann zu Text, grafischen Darstellungen, Videos und Tonaufzeichnungen sowie Objekten der Umwelt hinzugefügt werden. Die Form einer Annotation kann visuell, auditiv und taktil (z.B. bei blinden Menschen) sein. Eine Annotation hat den Zweck entweder Informationen zu strukturieren und festzuhalten, eine tiefer gehende Verarbeitung zur Problemlösung zu unterstützen oder eine Kommunikation zwischen Menschen zu ermöglichen.

Während Annotationen Ergänzungen bzw. Umformungen darstellen, sind Notizen selbständige Wissensartefakte. Eine Notiz kann somit ohne eine weitere Repräsentation funktionieren.

Zudem unterscheidet Riedewald [2003] zwischen einer Markierung und einer Annotation. Eine Markierung kann eine Annotation sein, wenn sie eine entsprechende Permanenz vorweist. Verwendet man beispielsweise beim Lesen einer Textstelle den Finger zum Markieren der gelesenen Zeile, liegt keine Annotation vor, da die Markierung nach dem Lesen der Zeile nicht bestehen bleibt. Hier ist eher von einer begleitenden Geste auszugehen. Eine Markierung ist somit eine temporäre Annotation, die nur für kurze Zeit existiert.

5.2 Einsatzmöglichkeiten von Annotationssystemen

Eine große Bedeutung erfuhr das Themenfeld des Annotierens im Zuge der breiten Einführung der EDV in der Arbeitswelt. Das Scheitern des papierlosen Büros wurde u.a. dem Umstand geschuldet, dass Menschen Dokumente lieber ausdrucken und auf Papier lesen und kommentieren als diese am Bildschirm zu lesen. Als eine der Ursachen wurde genannt, dass sich Annotationen auf Papier unkompliziert notieren lassen. Der Grund dafür sind die Vorteile, die Papierdokumente gegenüber elektronischen Dokumenten haben. Es handelt es sich dabei um ganz allgemeine, praktische Aspekte, wie Robustheit, Kompaktheit und Kompatibilität. Vergleichende Untersuchungen zum Lesen am Bildschirm und von Papier identifizierten darüber hinaus vor allem die besseren Möglichkeiten, in Dokumenten zu navigieren, Dokumente übersichtlich anzuordnen sowie insbesondere, sie zu annotieren (vgl. [O'Hara und Sellen, 1997]).

Aus dieser Motivation heraus wurden zahlreiche Entwicklungen von Software-Produkten, so genannten Annotationssystemen, angestoßen, um Lesern ein einfaches Annotieren von elektronischen Dokumenten zu ermöglichen und das Lesen und Lernen zu fördern (vgl. [Cadiz et al., 2000; Catlin et al., 1989; Chin-Yeh und Gwo-Dong, 2004; Schilit et al., 1998]). Parallel hierzu wurden Annotationssysteme als Werkzeug zum kollaborativen Schreiben von Textdokumenten entwickelt, um mehreren Koautoren das synchrone bzw. asynchrone Kommentieren und Diskutieren eines Textes zu ermöglichen (vgl. [Noël und Robert, 2003; Sharples et al., 1993; Zheng et al., 2006]). Eine ganz andere Motivation lag den Arbeiten von Marshall und Brush [2000; 2002; 2004] zugrunde. In ihren Arbeiten untersuchten sie die Nützlichkeit von persönlichen Annotationen von Studenten in Büchern und stellten sich der Frage, ob persönliche Annotationen hilfreich für andere Leser sein könnten. Mit der Einführung von Grafik-Tabletts und Tablet-PCs fanden zudem Untersuchungen darüber statt, ob Annotationen als Steuerungsbefehle für interaktive Software-Produkte genutzt werden können (vgl. [Gould und Salaun, 1987; Liao et al., 2008]).

Die Einsatzmöglichkeiten von Annotationssystemen sind somit vielfältig. Sie unterscheiden sich nach dem eingesetzten Werkzeug, dem Ort der Annotation sowie der Art und Weise, wie die Annotation ausgeformt wird (vgl. [Marshall, 1998; Moreland et al., 1997; Ovsiannikov et al., 1999]). „Markieren“, „Randnotizen einfügen“ und „Unterstreichen“ sind Beispiele für typische Annotationstätigkeiten beim aktiven Lesen (vgl. [Adler und van Doren, 1972; Schilit et al., 1998]). Sie erweisen sich auch als Kommunikationsmittel bei der Korrektur und Diskussion von Dokumenten als sehr nützlich (vgl. [Marshall, 1998; Neuwirth et al., 1990; Ovsiannikov et al., 1999]). Untersuchungen zeigten zudem, dass Annotationen beim kollaborativen Lernen (vgl. [Kienle, 2003; Moreland et al., 1997]) und bei der Lösung von Problemen (vgl. [Riedewald, 2003]) von großer Hilfe sein können. Zuletzt dienen Annotationen auch zum Strukturieren und Organisieren von Dokumenten. Mit Annotationen können zum einen die Inhalte von Dokumenten strukturiert und zum anderen eine Sammlung von Dokumenten organisiert und Beziehungen zwischen Dokumenten ausgedrückt werden. Diese eher technisch geprägte Motivation entstand aus der Notwendigkeit heraus, ausgewählte Bereiche u.a. von Text-, Audio-, Video und Geodaten elektronisch beschriften und verwalten zu können. Inhalte können so mit wenig Zeitaufwand wiedergefunden und Zusammenhänge zwischen Dokumenten erfasst werden (vgl. [Dmitriev et al., 2006; Fogli et al., 2004; Gemmell et al., 2002]).

5.2.1 Annotationssysteme zum aktiven Lesen

Das aktive Lesen nach Adler und van Doren [1972] ist eine Methode zur schnellen und effektiven Erfassung von Texten. Sie zeichnet sich dadurch aus, dass beim Lesen wichtige Textstellen unterstrichen und bei besonders wichtigen Textstellen auch am Seitenrand zusätzliche Markierungen (z.B. Pfeile, etc.) notiert werden. Hierdurch soll es möglich sein, den Text nicht noch einmal komplett lesen zu müssen, sondern die Annotationen als Lese- und Navigationshilfe verwenden zu können. Fragwürdige Inhalte werden „unterschlängelt“ und Beispiele am Rand markiert, Unklarheiten durch Fragezeigen und wichtige Aussagen durch Ausrufezeichen markiert. Pfeile dienen der Verdeutlichung von Textzusammenhängen. Neben komplizierten Abschnitten soll eine kurze stichpunktartige Zusammenfassung verfasst werden. Außerdem sollen die wichtigsten Gedanken, die beim Lesen entstehen, an den entsprechenden Textstellen aufgeschrieben werden. Beim aktiven Lesen stellt das Annotieren einen integralen Bestandteil dar und ist insbesondere im Umgang mit Fachtexten von essentieller Bedeutung (vgl. [Schilit et al., 1998]). Durch das Annotieren beim Lesen kann sowohl die Erfassung und das Verstehen des Textes unterstützt als auch eine spätere (Weiter-)Verarbeitung vorbereitet werden. Annotationen steuern durch ihre optische Hervorhebung die Aufmerksamkeit und fördern den Lesefortschritt (vgl. [Marshall, 1998]). Untersuchungen von Cioffi [1986] und Fass und Schumacher [1978] konnten nachweisen, dass Information aus einem gelesenen Text besser im Gedächtnis bleiben, wenn im Text annotiert wird.

Elektronische Annotationssysteme finden sich in Textverarbeitungsprogrammen wie z.B. Adobe Acrobat. Außerdem verfügen E-Book-Lesegeräte und Tablet-PCs über Funktionen zum Anbringen von Annotationen. Auch zahlreiche E-Learning-Plattformen wie z.B. die Open-Source-Software Moodle bieten Funktionen zum Annotieren von webbasierten Textdokumenten an.

5.2.2 Annotationssysteme zum Kommunizieren

Annotationen erfüllen zudem eine Kommunikationsfunktion. Sie werden häufig bei der Diskussion und Korrektur von textlichen und grafischen Artefakten genutzt, z.B. bei einer kollaborativen Autorenschaft, bei redaktionellen Korrekturen, auf Konstruktionszeichnungen und auch beim Paper Prototyping. Texte und Konzepte können direkt im Dokument annotiert und an weitere Personen übermittelt werden. Wird hierfür ein elektronisches Annotationssystem verwendet, können die annotierten Kommentare allen Mitgliedern einer Gemeinschaft zugänglich gemacht und zur Diskussion gestellt werden.

Elektronische Annotationssysteme bieten hierfür Funktionen zum Einfügen und Verwalten von Annotationen und zur Benachrichtigung anderer Personen. Durch die Verwendung von Annotationen kann jedes Gruppenmitglied an der Diskussion teilnehmen und dabei gleichzeitig über den Kontext der Materialien, über die diskutiert wird, verfügen. Ein Beispiel für eine solche Kommunikationsplattform stellt die webbasierte Kollaborationsplattform „Basic Support for Cooperative Work (BSCW)“[3]. Auch in Textverarbeitungsprogrammen wie Microsoft Word und Adobe Acrobat sind solche Funktionen integriert. Mit Bildschirmfotoprogrammen („Snapshot-Tools“) können auf der grafischen Benutzeroberfläche von Software-Produkten Kommentare annotiert werden. Einen ähnlichen Weg schlagen

[3] Fraunhofer BSCW, http://www.bscw.de, abgerufen am 30.11.2009

Naghsh [2005] mit dem Werkzeug Gabbeh und Glukhova [2009] mit dem Werkzeug Bubble Annotation Tool (BAT) ein. Sie entwickelten elektronische Annotationssysteme zur Erfassung von Verbesserungsvorschlägen zu Software von Anwendern.

5.2.3 Annotationssysteme zur Strukturierung und Organisation

Annotationen werden darüber hinaus auch in der Organisation und Strukturierung von Dokumenten bzw. Objekten angewendet. Ein typisches Beispiel hierfür ist die Beschriftung von Büchern in Bibliotheken oder die Beschriftung von Inventargegenständen in Unternehmen. Im privaten Alltag benutzt man unbewusst Annotationssysteme wenn man beispielsweise die Musiksammlung beschriftet. Riedewald [2003] spricht auch von einer Annotation, wenn Menschen die Mülltüte vor die Tür legen, um sich beim Verlassen der Wohnung an das „Müll rausbringen“ zu erinnern.

Der Prozess des Annotierens findet bei der Anbringung von Meta-Daten zu einem Objekt statt. Bei Annotationssystemen wird dies auch als semantisches Annotieren bezeichnet. In diesem Fall dienen die Annotationen zur Verbesserung der Maschinenlesbarkeit der Objekte. Mithilfe der Annotation kann eine Software die Bedeutung eines Objekts interpretieren und darauf aufbauend weitere Aktionen anstoßen. Eine Einsatzmöglichkeit des Annotierens findet sich in der Strukturierung, z.B. zur Bearbeitung und Archivierung von Text, Video- und Fotomaterial. In der Videobearbeitung werden Annotationen zum Erstellen eines Video-Index verwendet, so dass spezielle Zeitbereiche im Video mit Schlüsselwörtern markiert und bei einer späteren Suche wiedergefunden werden können. Annotationen können in Form von Schlüsselwörtern auch auf speziellen Bereichen eines Fotos bzw. auf dem gesamten Foto angebracht werden, um ein Foto zu charakterisieren und es später bei der Suche im Fotoarchiv zu indizieren. Eine besondere Art der Annotation von Fotomaterial stellt die geografische Annotation dar. Hier werden geografische Daten annotiert, um so Landkarten mit Schlüsselwörtern zu versehen und Kartenbereiche zu markieren. Dass Text dabei nicht die einzige Form zur Beschreibung einer Annotation sein muss, zeigt das von Walter und Nagypal [2008] entwickelte Werkzeug „ImageNotion“. Mit ImageNotion können Bilder zur Annotation von Bildern verwendet werden, um die Arbeit für Fotoredaktionen zu vereinfachen. Ferner lassen sich Annotationen auch zur Beschreibung von Beziehungen zwischen Dokumenten verwenden, wie es beispielsweise von Bottoni et al. [2003; 2004] mit dem Werkzeug „MADCOW“ entwickelt wurde.

5.2.4 Annotationssysteme zum kollaborativen Lernen

Während das aktive Lesen mithilfe von Annotationen den persönlichen Lernprozess eines Individuums unterstützt, können durch den Austausch von Annotationen mehrere Individuen gemeinsam an Lerninhalten und ergänzenden Dokumenten arbeiten, ihre Notizen anderen zur Verfügung stellen und mit anderen über die Inhalte und Annotationen diskutieren (vgl. [Davis und Huttenlocher, 1995; Nokelainen et al., 2004]). Dieser Themenbereich fand durch die Verbreitung des Internets und später durch den Markterfolg mobiler Geräte steigende Bedeutung, da hierdurch auch die Anwendungsmöglichkeiten stiegen. Durch das Internet wurde die Infrastruktur für den Austausch von Informationen bereitgestellt, so dass Lernsysteme ohne zusätzliche Kosten im Internet angeboten und von vielen Personen genutzt werden konnten. Bei den mobilen Geräten bestand die Überlegung, dass elektronische Dokumente ähnlich flexibel wie Bücher und Papier handhabbar sind und so auch das mobile Lesen und Annotieren von elektronischen Dokumenten möglich wird.

Untersuchungen von Marshall [1997; 1998] zeigen auf, dass potentielle Käufer von Büchern aus einem Universitätsantiquariat vor allem diejenigen Bücher bevorzugten, die wenige Markierungen oder Hervorhebungen und viele lange satzartige Kommentare enthalten. So wurde angemerkt, dass Markierungen bzw. Hervorhebungen das Lesen erschweren, da sie die Aufmerksamkeit durch den Text leiten. Neuen Lesern fällt es somit schwer, den markierten Text unvoreingenommen zu lesen. Außerdem können Markierungen störend auf das Erscheinungsbild der Buchseiten wirken. Anmerkungen in Form von längeren Sätzen hingegen sind für die Käufer interessant. Die Interpretationen und Hinweise der vorherigen Leser könnten möglicherweise auf vormals unbekanntes Wissen oder Material hinweisen. Diese Art von Annotationen stellt somit neues Wissen als Ergänzung zum Buch dar und erweist sich für den Leser als interessant und nützlich.

Die Forschung zu diesem Themenbereich ist stark werkzeugzentriert. Mithilfe von Befragungen und der Beobachtung des Annotationsverhaltens von Menschen (überwiegend im akademischen Umfeld, u.a. Studenten, Dozenten) wurden Anforderungen an elektronische Unterstützungswerkzeuge erarbeitet und darauf aufbauend Prototypen implementiert und evaluiert. Als Anforderungen wurden u.a. identifiziert, dass Annotationen einfach und schnell einzugeben sind, die Sichtbarkeit der Annotationen zwischen privat und öffentlich wählbar ist, Annotationen bzw. Kommentare zu Annotationen abzugeben sind und die Annotationen im Text bzw. auf Grafiken eingebettet angezeigt werden, ohne aber den Textfluss zu stören. Systeme, die das Annotieren zum kollaborativen Lernen ermöglichen, sind u.a. Notepal von Davis et al. [1999], ComMentor von Röscheisen et al. [1994; 1995], Hyperwave von Maurer [1996], Group Annotation Transducer (GrAnT) von Schickler et al. [1996], CoNote von Davis und Huttenlocher [1995]) Kukakuka von Suthers und Xu [2002], CritLink von Yee [2002], EDUCOSM von Nokelainen et al. [2003; 2004] sowie WebAnn und EPost von Brush et al. [2002].

5.2.5 Annotationssysteme zum Problemlösen

Annotationssysteme eignen sich nach Riedewald [2003] bei bestimmten Aufgabenstellungen zum Problemlösen. So können Annotationen als Strategie zur Reduktion der Arbeitsgedächtnisbelastung eingesetzt werden. Sie helfen dabei, die Komplexität einer externen Repräsentation zu strukturieren und in kleine Einheiten einzuteilen. Annotationen können ebenfalls dafür eingesetzt werden, die Ergebnisse von Zwischenschritten festzuhalten. So sind Annotationen bei solchen Aufgaben hilfreich, die in einzelne Schritte unterteilt werden können und nicht aus dem Gedächtnis heraus zu lösen sind.

Eine Form von Annotationen stellen Steuerungscodes (vgl. [Weidenmann, 1994]) dar, mit denen eine Abbildung für eine bessere Verständlichkeit modifiziert wird. Implizite Steuerungscodes stellen eine Umformung der Abbildung dar, ohne jedoch zusätzliche Informationen hinzuzufügen. Beispielsweise kann die Vergrößerung einer Abbildung oder die Komposition von mehreren Abbildungen annotiert werden. Mit expliziten Steuerungscodes werden spezielle Zeichenelemente, z.B. Pfeile, Rahmen, etc. zur Abbildung hinzugefügt. Ein weiteres Konzept in diesem Zusammenhang stellt die sekundäre Notation (vgl. [Green und Petre, 1996; Petre und Green, 1993]) dar, bei der anhand der Text- bzw. Bildorganisation (z.B. Größe, Nähe, Gruppierung, etc.) und der Verwendung von Zeichen, die von der formalen Syntax abweichen (z.B. Hervorhebungen, Kommentare, etc.), ergänzende Informationen zum eigentlichen Text- und Bildinhalt hinzugefügt werden. Beim Lösen von Problemen hilft die sekundäre Notation dabei, Informationen zu verdeutlichen (z.B. Struktur, Funktion oder Beziehungen) und das Lesen zu erleichtern.

Studien (vgl. [Cox und Brna, 1995; Hegarty und Steinhoff, 1997; Hegarty und Kozhevnikov, 1999; Larkin und Simon, 1987; Larkin, 1989]) ergaben, dass bei schwereren Aufgaben häufiger als bei leichteren annotiert wurde. Je schwieriger eine Aufgabe war, also z.B. je länger die erforderliche Schlussfolgerungskette war, desto öfter wurde annotiert. Allerdings konnte festgestellt werden, dass zu Beginn der Bearbeitung einer Aufgabe erst in Erwägung gezogen wird, ob sich der Einsatz von Annotationen lohnt. Außerdem konnten die Aufgabentypen, die einer Annotationsstrategie bedurften, nicht eindeutig klassifiziert werden, da die Teilnehmer ihre Lösungsstrategien unterschiedlich wählten.

Alternativ zu Annotationen können auch Notizen verwendet werden, um komplexe Sachverhalte zusammenzufassen (vgl. [Trafton und Trickett, 2001]). Im Unterschied zu Annotationen werden Notizen allerdings separat vom Dokument gehalten.

5.3 Anforderungen an Annotationssysteme

Erste Anforderungen an Annotationssysteme werden von Marshall [1997; 1998] und O'Hara und Sellen [1997] im Zusammenhang mit Annotationssystemen für digitale Bibliotheken beschrieben. In Interviews von Bibliotheksmitarbeitern und in Analysen von annotierten Büchern identifizierten sie eine Vielzahl an Anforderungen an Annotationssysteme:

- Während des Lesens möchten Leser ihre Annotationen direkt auf den Dokumenten und so nah wie möglich zum Bezugspunkt notieren.
- Die Annotationen müssen sich dabei vom umgebenden Dokument abheben, damit sie klar erkennbar sind.
- Zu berücksichtigen ist, dass Annotationen auch nur von temporärer Bedeutung sein können, wenn sie der Unterstützung beim Lesen oder Problemlösen dienen (vgl. auch [Riedewald, 2003]).
- Beim Lesen verwenden Leser häufig eigene individuelle Kodierungssysteme (z.B. unterschiedliche Farben und Formen) für die Unterscheidung der Semantik ihrer Annotationen. Die Leser gestalten ihre Kodierungssysteme dabei so, dass sie einen geringen Aufwand zum Annotieren und geringes Erinnerungsvermögen zum Merken der semantischen Strukturierung benötigen.
- Da manche Annotationen auch für andere Leser interessant sein könnten, sollten private Annotationen auch mit Einverständnis des Lesers öffentlich zur Verfügung gestellt werden können.
- Die Aktivität des Annotierens sollte in das Lesen integriert sein. Nach dem Vorbild des Papiers möchte der Leser durch das Annotieren so wenig wie möglich beim Lesen gestört werden. Eine Unterbrechung des Lesens wird nicht geduldet.

Eine nicht eindeutige Stellungnahme hinterlassen Marshall [1997; 1998] sowie O'Hara und Sellen [1997] bei der Diskussion, ob der Leser beim Annotieren freie Formen zeichnen darf oder zeitsparend aus einer Palette an vorgegebenen Annotationselementen auswählt. Einig sind sie sich darin, dass die

Annotationsaktivität so wenig Aufwand wie möglich machen sollte. Während das Annotieren auf Papier nur sehr geringen Aufwand kostet und höchste Flexibilität bietet, stellt das Annotieren am Computer eine größere Hürde dar, wenn zu einer Markierung gleichzeitig Text über die Tastatur eingegeben und Markierungen mit einem Zeigegerät erstellt werden sollen. Allerdings lassen sie offen, wie eine Lösung mit einem Annotationssystem aussieht. Abhilfe könnte nach Riedewald [2003] durch die Vorgabe einer großen Vielzahl unterschiedlicher Annotationsformen und einer hohen Bedienbarkeit geschaffen werden. Alternativ schlägt Riedewald [2003] die Verwendung eines Grafiktabletts oder eines interaktiven Bildschirms mit einem elektronischen Stift vor.

Ovsiannikov et al. [1999] führten eine umfassende Analyse von siebzehn existierenden Annotationssystemen durch und erfassten deren Funktionsangebot. Die untersuchten Annotationssysteme dienten zur Kommunikation (z.B. Microsoft Word), zum aktiven Lesen (z.B. XLibris), zum kollaborativen Lernen (z.B. HyperNews) sowie zum Strukturieren und Organisieren (z.B. Walden's Pat, Endnotes). Allerdings schweigen sich Ovsiannikov et al. [1999] darüber aus, welche der Funktionen für Akzeptanz und Produktivitätssteigerung von Relevanz sind. Vielmehr geben sie eine Übersicht an Möglichkeiten und Empfehlungen, wie Annotationssysteme gestaltet werden können. Ergänzend zu den oben genannten Anforderungen konnten sie in den einzelnen Systemen folgendes Funktionsangebot identifizieren:

- Grundsätzlich bieten Annotationssysteme die Möglichkeit, Annotationen entweder in Dokumenten, Multimediainhalten oder Internetseiten zu erstellen.
    - In Dokumenten können Annotationen zum Text und allen umgebenden Objekten platziert werden.
    - Multimediainhalte können in Form von Grafiken sowie Video- und Audioaufzeichnungen annotiert werden. Bei Grafiken können das gesamte Grafikdokument oder auch nur einzelne Bereiche der Grafik annotiert werden. Auf ähnliche Weise können bei Video- und Audioaufzeichnungen das Gesamtdokument oder einzelne zeitliche Bereiche der Aufnahme annotiert werden.
    - Auf Internetseiten können eine Seite oder einzelne Textbereiche und Grafikobjekte einer Seite annotiert werden.
- Als Inhalt einer Annotation dienen entweder Text, Links, Markierungen bzw. grafische Symbole, Zeichnungen oder Multimediainhalte (z.B. Sprachaufzeichnungen, Videoaufnahmen). Mit dem System „Dynomite" von Wilcox et al. [1997] ist es darüber hinaus möglich handschriftliche Notizen über einen Stift auf einem Tablet-PCs als Annotationen einzugeben.
- Die Speicherung der Annotationen erfolgt in Form von Metadaten entweder in den annotierten Inhalten bzw. Dateien direkt oder in einer externen Datenbank.
- Die Annotationen werden entweder direkt bei den annotierten Inhalten platziert (z.B. direkt im Text oder am Rand eines Dokuments eingebettet) oder separat in einem gesonderten Bereich aufgelistet. In dem gesonderten Bereich können die Annotationen in einer separaten Liste übersichtlich dargestellt werden. In dieser Liste ist es möglich, zwischen den Annotationen zu navigieren und Annotationen zu bearbeiten.

- Einige Annotationssysteme verfügen über eine Stichwortsuche zur Suche nach Annotationen.
- Mehrere Annotationssysteme unterstützen den Austausch und die Diskussion von Annotationen. Annotationen können entweder als öffentlich markiert oder bestimmten Personen bzw. Gruppen zugänglich gemacht werden. Zu den Annotationen können zudem Diskussionen geführt bzw. Annotationen zu Annotationen gesetzt werden. Die technische Realisierung erfolgt entweder über eine Import-Funktion zum Importieren von Annotationen aus annotierten Dokumenten oder über eine externe Datenbank.
- Einige der Systeme können mit unterschiedlichen Datenformaten umgehen und auf unterschiedlichen Plattformen betrieben werden.

5.4 Überblick über Methoden mit Annotationssystemen zur Anwenderbeteiligung

In der Literatur finden sich mehrere Ansätze für Methoden zur Anwenderbeteiligung, bei denen Annotationssysteme eingesetzt werden. Dies sind u.a. Arbeiten von Moore und Shipman [2000] zur Erstellung von grafischen Benutzeroberflächen mit dem „Graphical Requirements Collector“, von Naghsh et al. [2005] mit „Gabbeh“ zur Kommentierung von elektronischen Prototypen mit grafischen Annotationen, von Dörner et al. [2008] mit „Infrastructure Probes“ als ethnografische Methode, von Lohmann et al. [2008] mit „Softfox” zur Kommentierung von Webseiten und von Glukhova et al. [2009] mit „Bubble Annotation System“ zum Annotieren von Bildern über einen Videostream (MPEG-7). Im Folgenden werden diese Arbeiten näher beschrieben, um einen Überblick über ihre Ansätze zu geben und anschließend Vor- und Nachteile von Annotationssystemen zur asynchronen Anwenderbeteiligung zu diskutieren.

„Graphical Requirements Collector” (GRC) von Moore et al. [2000; 2003] dient der Erhebung von Anforderungen mit Hilfe der Erstellung von grafischen Benutzeroberflächen. Anwender können mit GRC funktionslose grafische Benutzeroberflächen erstellen und textliche Beschreibungen zu den grafischen Elementen hinzufügen (vgl. Abbildung 5.1). Sie können darüber hinaus auch Beziehungen bzw. Relationen zwischen den grafischen Elementen definieren. Zur Analyse der Anwendervorschläge können sich die Entwickler die GRC Artefakte betrachten und Rückschlüsse für die weitere Entwicklung treffen. Außerdem kommt eine Komponente zur Analyse von natürlich-sprachlichen Texten („Natural Language Processing (NLP) Approach“) zum Einsatz, mit der die gewählten Begrifflichkeiten der Anwender ausgewertet, Ähnlichkeiten zwischen den Texten der Anwender identifiziert und die relevanten Begriffe der Domäne der Anwender extrahiert werden.

Moore [2000] hebt ausdrücklich hervor, dass die mit GRC erstellten grafischen Benutzeroberflächen nur als Kommunikationsartefakte über die Anforderungen der Anwender genutzt werden und nicht als Vorschläge für die grafische Gestaltung der Software herangezogen werden können. So sollen die Anwender mit wenig Aufwand ihre Wünsche auf den Bildschirm platzieren und sich keine Gedanken über die Gestaltung machen. Ihre Evaluation im Vergleich mit einem textbasierten Fragebogen ergab, dass die Studienteilnehmer mit dem grafischen Editor eine höhere Anzahl von Anwenderbeiträgen erstellten. Die Konstruktivität war ebenfalls höher. Allerdings verbrachten die Studienteilnehmer auch mehr Zeit mit dem grafischen Editor.

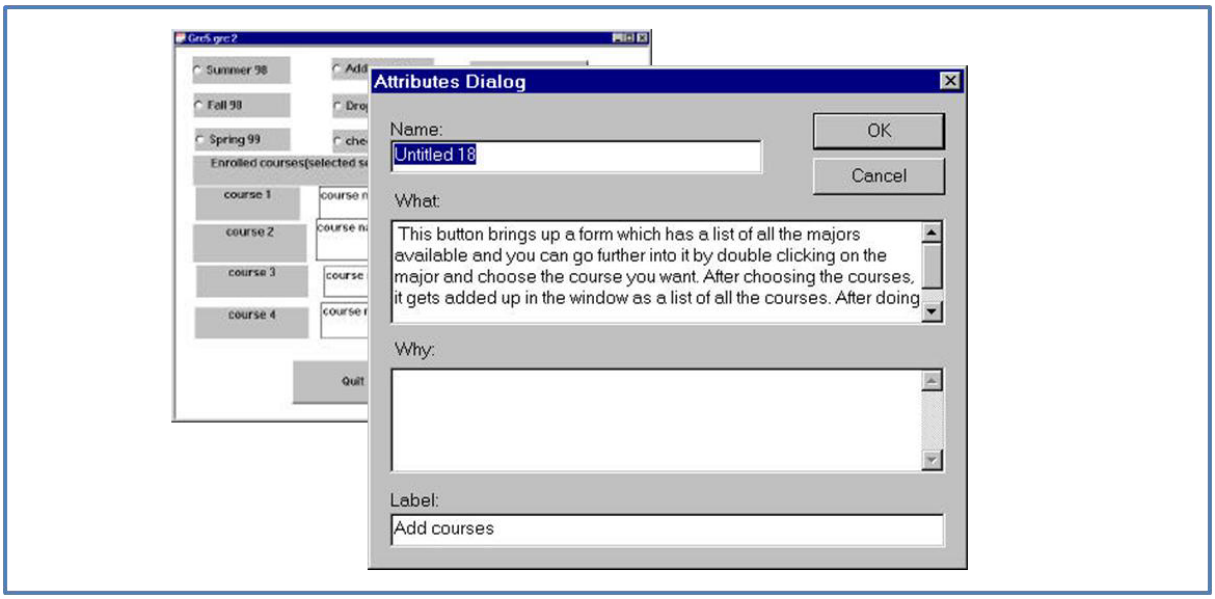

Abbildung 5.1: GRC beim Erstellen einer Benutzeroberfläche [Moore, 2003]

Mit „Gabbeh“ verfolgen Naghsh et al. [2004; 2005] das Ziel, die elektronische Variante des Papier-Prototypen, den sogenannten elektronischen Papier-Prototypen, um Funktionen zur Abgabe und Diskussion von Verbesserungsvorschlägen zu erweitern. Werkzeuge für elektronische Papier-Prototypen sind u.a. DENIM (vgl. [Lin et al., 2002]) und Freeform (vgl. [Plimmer und Apperley, 2003]). Mit diesen ist es möglich, Entwürfe für Papier-Prototypen am PC zu erstellen und Usabilty Tests mithilfe von Papierausdrucken einzusetzen. Mit Gabbeh sollen die Papier-Prototypen in elektronischer Form getestet werden können, so dass die Anwender die Prototypen am PC testen und kommentieren können. Kommentare zu den Elementen des Prototypen können in Form von Annotationen hinzugefügt werden (vgl. Abbildung 5.2).

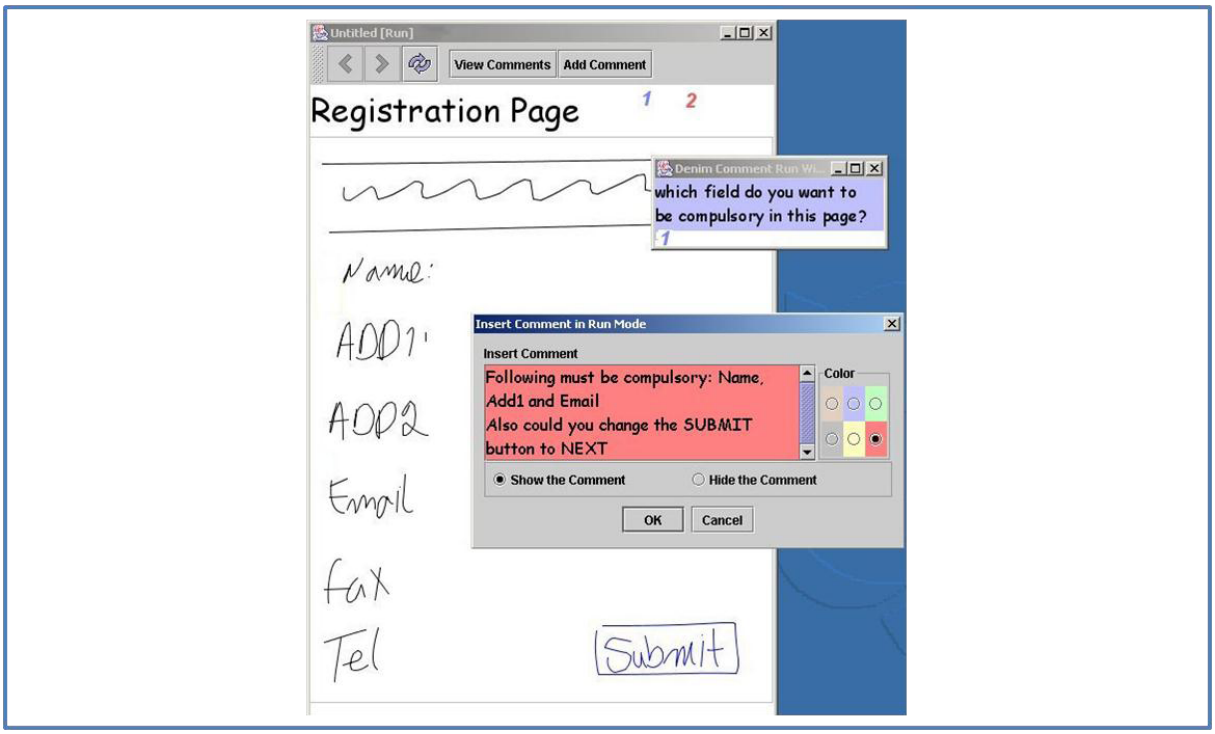

Abbildung 5.2: Gabbeh beim Einfügen eines Kommentars [Naghsh und Dearden, 2004]

Der Ansatz der „Infrastructure Probes“ (IP) von Dörner et al. [2008] stellt eine ethnografische Methode zur Analyse des Arbeitskontextes des Anwenders dar. Mit IP soll dem Umstand Rechnung getragen werden, dass ethnografische Methoden einen hohen Zeitaufwand für die Beobachtung der Anwender benötigen. Daher soll sich der Anwender mit IP selbst beobachten und seinen Arbeitskontext dokumentieren. Dabei kommt ein ganzes Bündel an Werkzeugen (vgl. Abbildung 5.3), u.a. Bildschirmfotoprogramm, Digitalkamera, Notizzettel, Internet-Tagebuch, Feedback-Formulare und ein Notizbuch, zum Einsatz. Der Anwender kann damit während seiner Arbeit protokollieren, wie sich die Nutzung der Software in den Arbeitsalltag integriert und an welchen Stellen Verbesserungsbedarf besteht. Dem Anwender steht dabei frei, wie und wann er die Werkzeuge nutzt. Die Evaluation des Ansatzes ergab, dass vor allem Digitalkamera und Bildschirmfotoprogramm zum Einsatz kamen. Die Anwender bevorzugten die elektronischen Werkzeuge aufgrund ihrer effizienten Handhabung. Papier und Stift (Notizzettel, Notizbuch, Feedback-Formular) wurden abgelehnt.

Abbildung 5.3: Infrastructure Probe Paket [Dörner et al., 2008]

„Softfox“ wurde von Lohmann et al. [2008] als Erweiterungsmodul (Add-On) zum Internetbrowser Mozilla Firefox entwickelt. Mit diesem Ansatz (vgl. Abbildung 5.4) kann über eine Schaltfläche (1) in Firefox die aktuell besuchte Internetseite kommentiert werden. Über ein Formular in einem Pop-Up Fenster (2) können Titel (4), Beschreibung (3) und Schlagwörter (5) zum Feedback eingegeben werden. Optional kann der Anwender die Elemente der Internetseite, auf die er sich in seinem Feedback bezieht, selektieren (7a) und dadurch auf eine textbasierte Beschreibung der Elemente (7b), auf die er sich bezieht, verzichten. Um die mehrfache Abgabe desselben Vorschlages zu vermeiden, werden während der Texteingabe ähnlichen Einträgen angezeigt. Hierfür werden mittels computerlinguistischer Verfahren ähnlichen Einträge verglichen und dem Anwender die Einträge mit den meisten Überschneidungen angezeigt (6). Darüber hinaus werden Kontextinformationen automatisiert erfasst, welche die Auswertung der Anwenderbeiträge erleichtern sollen. Hierzu gehören die Adresse der Internetseite, Informationen aus dem HTTP-Header der geöffneten Webseite, die Größe des

Inhaltsbereichs des Firefox-Fensters, die Bildschirmauflösung, Standortinformationen zum Nutzer sowie Angaben zum verwendeten Webbrowser und Betriebssystem.

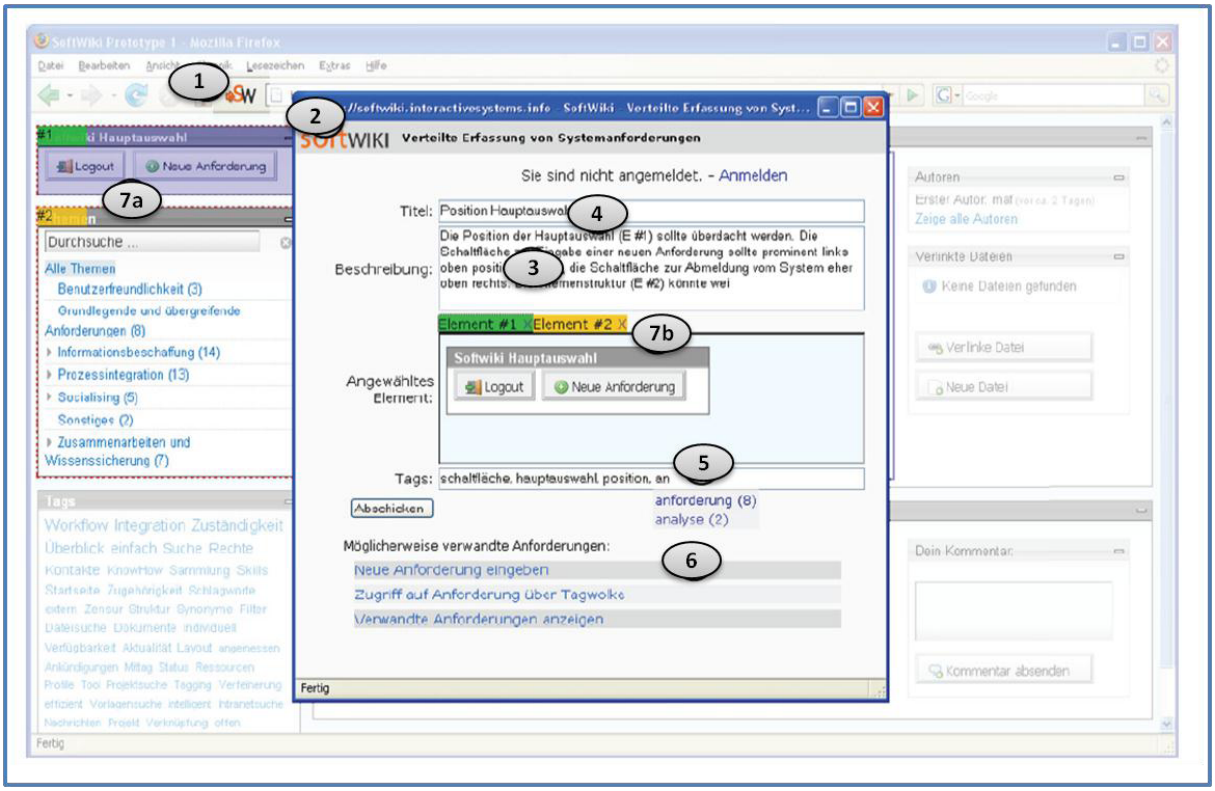

Abbildung 5.4: Softfox beim Einfügen eines Kommentars [Lohmann et al., 2008]

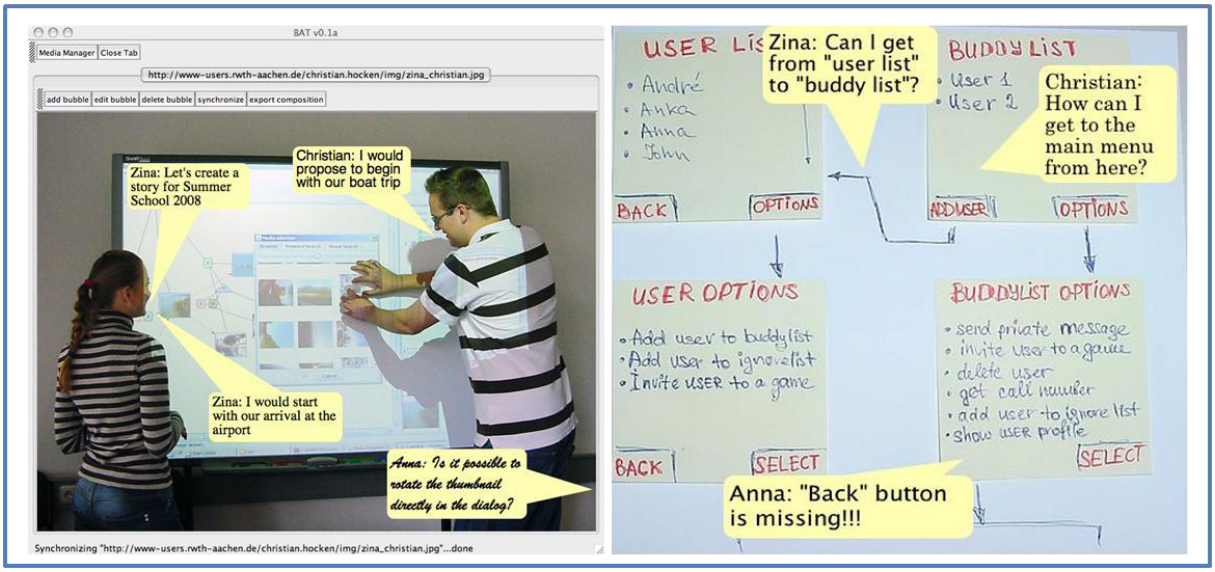

Abbildung 5.5: Interaktion (l.) und annotiertes Bild (r.) mit BAT [Glukhova et al., 2009]

Das von Glukhova et al. [2009] entwickelte „Bubble Annotation Tool“ (BAT) verfolgt das Ziel, Communities of Practice (CoP) bei der Erhebung von Software-Anforderungen zu unterstützen. An einem PC können mit Hilfe des BAT Annotationen in Form von Sprechblasen auf beliebige grafische Darstellungen (Grafiken, Bildschirmfotos, Fotos, Diagramme, etc.) platziert werden. Über die Anzeigekomponente können Bilder in unterschiedlichen Formaten zur Annotation eingestellt oder auf bereits

annotierte Bilder zugegriffen werden. Der Anwender kann sich die Bilder ansehen und Kommentare abgeben. Der Zugriff auf die Anzeigekomponente ist von jedem beliebigen PC aus möglich. In Abbildung 5.5 sind beispielhaft ein Bildschirmfoto von BAT und ein Ergebnisdokument dargestellt.

5.5 Vor- und Nachteile von Annotationssystemen zur asynchronen Anwenderbeteiligung

Aus den bisherigen Arbeiten zu Annotationen und Annotationssystemen lassen sich mehrere Hinweise auf Vor- und Nachteile des Einsatzes von Annotationssystemen zur asynchronen Anwenderbeteiligung ableiten. Dieser Betrachtung liegt das Szenario zugrunde, dass Anwender mithilfe eines Annotationssystems Anwenderbeiträge erstellen und Moderatoren, Entwickler und Entscheider diese Anwenderbeiträge einsehen und weiterverarbeiten können.

Folgende Vorteile können sich mit Annotationssystemen im Rahmen einer asynchronen Anwenderbeteiligung ergeben:

- Das Annotieren ist eine intuitive Vorgehensweise, die bereits beim aktiven Lesen und Problemlösen häufig eingesetzt wird. Es ist zu erwarten, dass Anwender diese Form der Kommunikation leicht erlernen und anwenden können. Die Ergebnisse aus den Evaluationen von IP und GRC zeigen auf, dass die Möglichkeit zum Annotieren von den Anwendern häufig genutzt wird. Sie ist schneller und einfacher anzuwenden als die Formulierung einer ausschließlich textlichen Beschreibung. Annotationssysteme führten bei ihren Untersuchungen auch zu einer höheren Anzahl und Konstruktivität der Anwenderbeiträge.

- Annotationen nehmen unmittelbar Bezug zur annotierten Stelle. Moderatoren, Entwickler und Entscheider können somit schnell den Kontext und Diskussionsgegentand des Anwenderbeitrags erfassen ohne den gesamten Beitrag betrachten zu müssen. Dies kann bei einer großen Menge an Anwenderbeiträgen hilfreich sein, um schnell eine Übersicht über die Anwenderbeiträge zu erlangen. Bei einer rein textlichen Beschreibung müsste der Text erst vollständig gelesen werden, um den Kontext zu erfassen.

- Annotationssysteme können für unterschiedliche Plattformen und unterschiedliche Arten von Software bzw. Prototypen eingesetzt werden.

- Der Einsatz von Prototypen in frühen Phasen eines Software-Projektes hat sich als ein wichtiger Erfolgsfaktor für eine effektive Anwenderbeteiligung erwiesen. Für den Einsatz von Annotationssystemen bedeutet dies, dass für sie auch immer ein Bezugssystem existiert, so dass Anwender mithilfe eines Annotationssystems zu den Elementen des Prototypen Annotationen erstellen können.

Allerdings ist insgesamt mit folgenden Nachteilen zu rechnen:

- Individuelle Kodierungssysteme können dazu führen, dass andere Personen als der Autor die Annotationen nicht nachvollziehen können. Hinzu kommt, dass Autoren beim Annotieren den geringstmöglichen Aufwand suchen und daher Kodierungssysteme bevorzugen, die kurz

und knapp sind. Daher sind Annotationen unter Umständen nicht allgemein verständlich beschrieben.

- Annotationssysteme beziehen sich auf ein Bezugssystem, d.h. einen Prototypen, eine grafische Darstellung oder einen Text. Dadurch könnte der Fokus der Anwender zu stark auf vorhandene Ideen gelegt werden. Neue Ideen und radikale Änderungen würden eher nicht entstehen.

- Für den Moderator sind Informationen zum Aufgabenumfeld und zu den Bedürfnissen der Anwender wichtig. Bei Annotationssystemen ist nicht zu erwarten, dass solche Informationen bereitgestellt werden. Stattdessen würden Anwender eher zur Abgabe von Kommentaren zu einem Prototypen motiviert als dass sie Informationen über ihren Kontext bereitstellen oder über ihre Arbeitsprozesse reflektieren. Annotationssysteme könnten Anwender zudem motivieren, sich ausschließlich auf funktionale Anforderungen zu beziehen. Es besteht die Gefahr, dass nicht-funktionale Anforderungen vernachlässigt werden. Daher ist zu befürchten, dass Annotationssysteme nur für einen begrenzten Bereich einsetzbar sind.

Abschließend lässt sich zusammenfassen, dass der Einsatz von Annotationssystemen zu einer Verbesserung der Konstruktivität und Anzahl der Anwenderbeiträge sowie zu einer Verringerung der Belastung der Anwender führen kann. Die Verständlichkeit der Anwenderbeiträge könnte jedoch verringert werden. Es ist zudem zu erwarten, dass Annotationssysteme zu stark auf dem Einsatz von Prototypen basieren und ihr Einsatzbereich dadurch reduziert ist. Neue bzw. radikale Ideen der Anwender werden eher nicht zu erwarten sein.

Darüber hinaus ist zu beachten, dass die bisherigen Untersuchungen zu Annotationssystemen zur Anwenderbeteiligung in Form von Experimenten mit kleinen Gruppen (weniger als zehn Teilnehmer) und vorwiegend bestehend aus Studenten und Kollegen durchgeführt wurden. Erfahrungen mit einem größeren Umfang von Teilnehmern und im realen Anwendungsfeld existieren bislang nicht.

5.6 Fazit zu Kapitel 5

Ziel von Kapitel 5 war es, die Potentiale von Annotationssystemen im Rahmen einer asynchronen Anwenderbeteiligung zu untersuchen und der Frage nach der Notwendigkeit einer neuen Methode zur asynchronen Anwenderbeteiligung nachzugehen, bei der Annotationssysteme zum Einsatz kommen. Dabei ist von zentraler Bedeutung, dass Annotationssysteme sowohl die Verständlichkeit, Anzahl und Konstruktivität der Anwenderbeiträge erhöhen als auch die Belastung der Anwender verringern um den Nachteilen von Methoden zur asynchronen Anwenderbeteiligung entgegenwirken zu können. Hierfür wurden Begriffe zu Annotationen und Annotationssystemen und ihre Einsatzmöglichkeiten aufgearbeitet, den aktuelle Stand der Forschung zu Annotationssystemen zusammengefasst sowie die Vor- und Nachteile für ihren Einsatz zur Beteiligung von Anwendern erörtert.

Dabei konnten folgende eigene Beiträge herausgearbeitet werden:

- Anhand einer Literaturanalyse wurden Anforderungen an Annotationssysteme erarbeitet und ihre unterschiedlichen Einsatzmöglichkeiten zusammengefasst. Annotationssysteme eignen sich demnach zum aktiven Lesen, zum Kommunizieren, zur Strukturierung und Organisation,

zum kollaborativen Lernen und zum Problemlösen. Für den Zweck der Anwenderbeteiligung kristallisierte sich vor allem ihre Kommunikationsfunktion als geeignetes Instrument für die Erfassung von Anwenderbeiträgen heraus.

- Darauf aufbauend wurden Vor- und Nachteile des Einsatzes von Annotationssystemen bei einer Anwenderbeteiligung erörtert. Demnach kann der Einsatz von Annotationssystemen zu einer Verbesserung der Konstruktivität und Anzahl der Anwenderbeiträge sowie zu einer Verringerung der Belastung der Anwender führen. Allerdings ist zu damit zu rechnen, dass sich die Verständlichkeit der Anwenderbeiträge verringert. Außerdem ist von einem begrenzten Einsatzbereich von Annotationssystemen auszugehen, da lediglich Meinungen zu Prototypen und somit weder Informationen zum Kontext des Anwenders noch neue bzw. radikale Ideen erfasst werden können.

In den beiden letzten Kapiteln 4 und 5 wurde somit die dritte Forschungsfrage dieser Arbeit nach den Potentialen und Grenzen von Methoden zur asynchronen Anwenderbeteiligung beantwortet. Hiermit endet der dritte Teil dieser Arbeit. Annotationssysteme besitzen das Potential, die Nachteile einer asynchronen Anwenderbeteiligung bis auf die Verständlichkeit der Anwenderbeiträge zu beseitigen. Daher ist die Notwendigkeit gegeben, eine neue Methode zur asynchronen Anwenderbeteiligung, bei der Annotationssysteme zum Einsatz kommen, zu entwickeln und zu evaluieren.

Im vierten und letzten Teil dieser Arbeit wird nun untersucht, wie eine solche Methode zu gestalten ist und welche Effekte sie auf die Effizienz der Entwicklung, die Qualität der Anwenderbeiträge und die Belastung der Anwender haben wird. Unter der Bezeichnungen OpenProposal wird im folgenden Kapitel 6 die neue Methode aufbauend auf den bisherigen Erkenntnissen konzipiert und in den nachfolgenden Kapiteln evaluiert.

6 OpenProposal: Eine neue Methode zur asynchronen Anwenderbeteiligung

In diesem Kapitel wird die Methode OpenProposal und das zugehörige Annotationssystem zur Unterstützung der Kommunikation zwischen Anwendern und Moderatoren vorgestellt. Diese Methode soll eine asynchrone Anwenderbeteiligung in Software-Projekten ermöglichen und die Effizienz der Entwicklung, die Qualität der Anwenderbeiträge und die Belastung der Anwender verbessern. Hierfür werden Erwartungen aus der Praxis erhoben, die Ziele der Methode und daraus resultierenden Lösungskonzepte für das OpenProposal Annotationssystem beschrieben, und anschließend in mehreren Iterationen Anforderungen definiert und überarbeitet. Darauf aufbauend wird das zugrunde liegende Prozessmodell von OpenProposal entworfen und die Einsatzmöglichkeiten in den unterschiedlichen Phasen von Software-Projekten werden erläutert.

Zur Vereinfachung der Schreibweise werden im Folgenden sowohl die Methode als auch das Annotationssystem unter der Bezeichnung OpenProposal geführt. Soweit nicht anders aufgeführt, beziehen sich methodische bzw. fachliche Aspekte auf die Methode und technische Aspekte auf das Annotationssystem.

6.1 Erwartungen aus der Praxis

Im Rahmen des Forschungsprojektes CollaBaWü[4] wurden u.a. Erwartungen der Praxis an Methoden zur Anwenderbeteiligung erhoben. Am 1. Juli 2004 startete das im Rahmen des Forschungsverbunds „PRIMIUM" des Landes Baden-Württemberg initiierte Forschungsprojekt „CollaBaWue", welches vom FZI Forschungszentrum Informatik und den Universitäten Karlsruhe und Mannheim sowie den sechs Anwenderunternehmen Badische Beamtenbank (BBank), Landesbank Baden-Württemberg (LBBW), Cellent, Struktur, Entory (2008 wurde die Entory in Cirquent umbenannt) und Excelsis ins Leben gerufen wurde. Das Projekt wurde durch das Wissenschaftsministerium des Landes Baden-Württemberg für drei Jahre gefördert. Gegenstand der interdisziplinären Forschungsaktivitäten in CollaBaWü war die kollaborative, komponentenbasierte Entwicklung von Unternehmenssoftware im Finanzdienstleistungsbereich von Baden-Württemberg mit der Zielsetzung der produktiveren Erstellung domänenspezifischer betrieblicher Anwendungen. Übergeordnetes Ziel des Projekts war eine fortschreitende „Industrialisierung" der Software-Entwicklung zur Stärkung des Wirtschaftsstandorts Baden-Württemberg bei der Erstellung von Unternehmenssoftware.

In zahlreichen Gesprächen und Analysen mit den Anwendungspartnern LBBW und BBBank fand in den Jahren 2005 und 2006 ein intensiver Austausch zum Thema Anforderungsmanagement und

[4] CollaBaWü, http://www.collabawue.de, abgerufen am 17.03.2009

Anwenderbeteiligung statt. Die Gespräche wurden in Form von narrativen Interviews nach Schnell et al. [2008] durchgeführt. Dabei wurden zu Beginn die Ziele des Forschungsprojektes und die Inhalte des Arbeitspakets „Kollaborative Software-Entwicklung“ vorgestellt. Anschließend fand eine Diskussion über die Situation im Unternehmen und Verbesserungsmöglichkeiten mithilfe kollaborativer Software-Entwicklungsmethoden und -werkzeugen statt. Die Gespräche wurden bei beiden Unternehmen aufgezeichnet und im Anschluss ausgewertet. Die Aufzeichnungen und die Ergebnisse der Auswertungen wurden mit den Gesprächspartnern abgestimmt.

Die Schwerpunkte der Untersuchungen bei der LBBW lagen im bereichsübergreifenden Anforderungsmanagement und in der Anwenderbeteiligung in Software-Projekten. Ansprechpartner auf Seiten der LBBW waren Mitarbeiter der Abteilung IT-Strategie, die für die strategische Planung der Systemlandschaft zuständig waren. Gemeinsam mit den Produktmanagern und dem Vorstand nahmen sie die Rolle des Entscheiders innerhalb der LBBW wahr. Als Anwender wurden die Mitarbeiter der Fachabteilungen verstanden. Zu den Entwicklern gehörten sowohl hausinterne Entwickler als auch externe Dienstleister, wobei die externen Dienstleister eine große Mehrheit darstellen. Die Rolle des Moderators erfüllten die Kundenmanager. Als Kunden wurden bei der LBBW die Anwender der Software-Systeme - also die Mitarbeiter der LBBW - verstanden. Die Kundenmanager waren dafür zuständig, die Probleme und Bedürfnisse der Anwender zu erfassen und Lösungsmöglichkeiten bereitzustellen.

Aus Sicht der LBBW ist es von großer Bedeutung, die Methodik des Anforderungsmanagements bei der LBBW im Zuge der Fusion mit anderen Landesbanken zu verbessern und damit die Konsolidierung der Systemlandschaft zu unterstützen. Der LBBW[5] stellten sich hierbei die üblichen Herausforderungen eines großen Unternehmens. Mit einer Bilanzsumme im Konzern von 443 Mrd. Euro (Stand: 31.12.2007), ca. 200 Filialen und weltweit 26 Stützpunkten sowie rund 12.250 Mitarbeitern zählt der LBBW-Konzern zu den fünf größten Kreditinstituten in Deutschland und zu den 50 größten Banken weltweit. Zudem übernahm die LBBW zum Anfang des Jahres 2005 die Landesbank Rheinland-Pfalz als 100-prozentige Tochter. Die Baden-Württembergische Bank (BW-Bank) gehört seit 01. August 2005 ebenfalls zur LBBW. In dieser Situation ist die strategische Planung der Systemlandschaft ein zentraler Bestandteil und bedarf umfangreicher Abstimmungsprozesse. Die Komplexität der Systeme stellt eine große Herausforderung dar, da zum einen die Systemlandschaft teilweise mit dem Sparkassen-Verbund gemeinsam entwickelt und betrieben wird und zum anderen die meisten Systemkomponenten von externen Dienstleistern bereit gestellt werden.

Aus den Erfahrungen der letzten Jahre heraus existiert der Wunsch, die Flexibilität und Agilität im Anforderungsmanagement zu verbessern und die Software-Planung den Bedürfnissen der Praxis zeitnah anpassen zu können. Von einer verstärkten Integration der Anwender in den Planungsprozess verspricht sich die Abteilung IT-Strategie eine Verbesserung des Planungshorizonts, um den Bedarf der Anwender frühzeitig einschätzen zu können. Außerdem erwarten sie eine höhere Akzeptanz bei der anschließenden Systemeinführung. Die bereits bestehenden Prozesse innerhalb der LBBW werden hierfür als nicht ausreichend bewertet.

[5] LBBW, http://www.lbbw.de, Jahresbericht 2008 der LBBW, abgerufen am 17.03.2009

Bei der LBBW existieren zum Zeitpunkt der Befragung bereits zahlreiche Möglichkeiten, sich als Anwender pro-aktiv einzubringen. Für die einzelnen Abteilungen der LBBW sind Kundenmanager zuständig, die sich um Software-Probleme der Anwender bei der LBBW kümmerten und Hilfestellung leisteten. Dabei sollen auch die Zufriedenheit der Anwender und der zukünftige Bedarf an weiterer Unterstützung erfasst werden. Diese Informationen werden an die Produktmanager der Softwareprodukte weitergeleitet und es werden Verbesserungsmöglichkeiten diskutiert. Darüber hinaus kann ein Anwender bei Software-Problemen den User-Support kontaktieren und sich bei eventuell auftretenden Problemen helfen lassen. In beiden Fällen wird der Mitarbeiter persönlich am Telefon, direkt vor Ort oder über eine Remote-Desktop-Verbindung beraten. Eine weitere Möglichkeit ist es, Vorschläge zu Software-Produkten über die Abteilungs- und Bereichsleitung einzureichen. Diese Vorschläge werden als Text-Dokumente an eine zentrale Einrichtung versendet, wo sie ausgewertet und im Rahmen sogenannter Releasekonferenzen zweimal im Jahr von Entscheidungsträgern diskutiert werden. Bei diesen Releasekonferenzen werden die nächsten Schritte bei Software-Projekten für das anstehende Halbjahr beschlossen. Außerdem existiert bei der LBBW ein betriebliches Vorschlagswesen mit einem monetären Belohnungssystem. Hier kann jeder Mitarbeiter jegliche Arten von Vorschlägen zur Verbesserung der Unternehmensprozesse einreichen. Sollten sich dadurch Einsparungen ergeben, wird der Mitarbeiter erfolgsabhängig belohnt.

Für den Zweck einer aktiven Anwenderbeteiligung werden die bisherigen Ansätze jedoch als zu formalistisch und aufwändig bewertet. Sie ersticken Anwenderinnovation in formalistischen Prozessen. Bisherige Vorschläge adressieren vor allem Probleme in der Betriebsorganisation. Vorschläge für die betrieblichen Informationssysteme werden nur sehr selten eingebracht. Daher wird eine Methode zur aktiven Anwenderbeteiligung erwünscht, mit der Mitarbeiter zur Mitgestaltung ihrer Arbeitsplatzsysteme und zur Abgabe von Verbesserungsvorschlägen motiviert werden. Dabei reicht aus ihrer Sicht das bisherige betriebliche Vorschlagswesen nicht aus, da es für die Abgabe von Vorschlägen zur Betriebssoftware nicht geeignet war. Gleichzeitig soll der Aufwand der Koordination der Anwenderbeteiligung gering gehalten werden. Dies soll durch eine effiziente Verwaltung der Vorschläge erreicht werden können, wobei vor allem Wert auf Verständlichkeit und Nachvollziehbarkeit der Vorschläge gelegt werden soll, um Missverständnisse und somit zusätzlichen Aufwand für die Kommunikation zu vermeiden.

Gleichzeitig soll ein solches Vorschlagssystem auch die bereichsübergreifende Zusammenarbeit verbessern und die Transparenz zwischen den einzelnen Abteilungen erhöhen. Anhand der Mitarbeitervorschläge aus den unterschiedlichen Abteilungen sollen bereichsübergreifende Synergien früher erkannt und genutzt werden können. Synergien können sich ergeben, wenn aus mehreren Abteilungen ähnliche Anwenderbeiträge für eine Software eingehen und die Umsetzung eines solchen Vorschlages gleichzeitig mehreren Abteilung zu Gute kommt.

Die Gespräche mit der BBBank erfolgten im selben Zeitraum. In diesen Gesprächen wurde aufgrund der Aufgabenstellung des Forschungsprojektes eine ähnliche Problemstellung diskutiert. Ansprechpartner auf Seiten der BBBank waren Mitarbeiter der Abteilung EDV / Organisation. Aufgrund der Organisationsform der Genossenschaft der BBBank und ihrer engen Partnerschaft mit den Volksbanken und Raiffeisenbanken wird die Software-Entwicklung und IT-Strategie zentral von der Fiducia AG

für alle Volksbanken, Raiffeisenbanken und ihre Partnern betrieben. Die BBBank eG[6] ist eine Genossenschaftsbank und mit 95 Filialen und 1.458 Mitarbeitern in acht deutschen Bundesländern vertreten. Die Bilanzsumme beträgt 6.552 Mio. € (Stand 31.12.2007). Dies bedeutet, dass die BBBank aus Kostengründen zum Zeitpunkt der Gespräche in sehr wenigen Fällen selbst Software-Entwicklung betreibt und ansonsten die gesamte Software-Entwicklung dem IT-Dienstleistungsunternehmen Fiducia IT AG[7] überlässt. Dies führt allerdings auch dazu, dass die BBBank sich bei Wünschen, die ihre Software betrafen, mit den anderen Volksbanken und Raiffeisenbanken abstimmen müssen. Das Anforderungsmanagement verläuft in bankenübergreifenden Arbeitsgruppen, welche aus Stellvertretern der Banken besteht. Jede Arbeitsgruppe beschäftigt sich mit einer bestimmten Produktkategorie. In diesen Arbeitsgruppen werden Änderungs- und Erweiterungsmöglichkeiten zu ihren Produktkategorien diskutiert und eine Liste an Verbesserungsvorschlägen an die Fiducia formuliert. Die Fiducia bewertet deren Machbarkeit und Wirtschaftlichkeit und sorgt dafür, dass die Anforderungen aus den Arbeitsgruppen sinnvoll in Einklang gebracht werden. Somit ist der Fiducia die Rolle des Moderators, Entscheiders und Entwicklers zuzuordnen. Die BBBank bzw. die Mitarbeiter in den Filialen der BBBank nehmen die Rolle der Anwender ein.

Die Befragung der Ansprechpartner der BBBank bestätigt die bei der LBBW erfassten Vorstellungen. Eine Anwenderbeteiligung in Software-Projekten ist erwünscht, da das Expertenwissen der Mitarbeiter, z.B. am Kundenschalter und am Beraterarbeitsplatz, als sehr hilfreich eingeschätzt wird. Mitarbeiterbefragungen werden allerdings selten durchgeführt, da ein Teil der Fiducia Mitarbeiter früher selbst im Bankenwesen gearbeitet hat und somit ein Verständnis für die Anwender bei den Entwicklern vorausgesetzt wird. In großen Projekten wird eine Auswahl von Mitarbeitern zu Beginn eines Projektes nach ihrer Arbeitsweise und Verbesserungsmöglichkeiten befragt und zum Ende eines Projektes zum Testen von Testinstallation gebeten. Zudem können Mitarbeiter über das betriebliche Vorschlagswesen Verbesserungsvorschläge einreichen, wobei anzumerken ist, dass Vorschläge zu Software selten auftreten. Das betriebliche Vorschlagswesen ist hierfür nicht geeignet, da ein mehrseitiges Formular mit Abschätzungen über Kosten und Nutzen ausgefüllt werden muss und hauptsächlich für Kosteneinsparungsmöglichkeiten in den Prozessabläufen vorgesehen ist. Daher ist eine Motivation der Anwender begrüßenswert, damit auch Vorschläge zu ihrer Software erstellt werden. Eine effiziente Verwaltung der Vorschläge wird ebenfalls vorausgesetzt. Die Erkennung bereichsübergreifender Synergien ist der BBBank ebenfalls ein wichtiges Anliegen und wird durch die Einrichtung eines bereichsübergreifenden Multiprojektmanagements angegangen. So sollen Synergien zwischen einzelnen Projekten besser erkannt und genutzt werden können.

Zusammengefasst lässt sich folgern, dass LBBW und BBBank trotz ihrer unterschiedlichen Ausrichtungen vergleichbare Erwartungen an Methoden zur Anwenderbeteiligung stellen:

- Methoden sollen Anwender zur Mitgestaltung ihrer Software motivieren, u.a. soll die Anzahl und die Qualität der Beiträge gesteigert werden. Außerdem wünscht man sich eine hohe Teilnehmerzahl um die Repräsentativität der Anwenderbeiträge sicherstellen zu können. Dies soll durch eine Vereinfachung der Erstellung von Anwenderbeiträgen erfolgen.

---

[6] BBBank, http://www.bbbank.de, Jahresbericht 2008 der BBBank, abgerufen am 17.03.2009

[7] Fidicua IT AG, http://www.fiducia.de, abgerufen am 17.03.2009

- Methoden sollen eine effiziente Verwaltung der Anwenderbeiträge ermöglichen. Es wird befürchtet, dass durch eine erhöhte Anzahl von Anwenderbeiträgen der Aufwand für ihre Bearbeitung steigt. Daher sollen Methoden diesen Umstand berücksichtigen und Lösungen bereitstellen, mit denen Anwenderbeiträge einfach und schnell bearbeitbar sind. Außerdem sollen die Anwenderbeiträge auch eine hohe Verständlichkeit aufweisen, so dass Entscheider und Entwickler ohne weitere Rückfragen die Vorschläge bewerten und umsetzen können. Einfache formale Strukturen wären hilfreich um dem Anwender eine Gliederung ihrer Anwenderbeiträge vorzugeben.

6.2 Ziele und Lösungskonzepte

Aus den vorherigen Kapiteln konnte die Erkenntnis gewonnen werden, dass bei der Entwicklung einer Methode zur asynchronen Anwenderbeteiligung die Effizienz der Entwicklung, die Qualität der Anwenderbeiträge und die Belastung der Anwender berücksichtigen werden soll. Der Schwerpunkt der Entwicklung liegt dabei auf der Qualität der Anwenderbeiträge und der Belastung der Anwender, die bislang bei einer asynchronen Anwenderbeteiligung ungünstig ausfielen.

Mit OpenProposal soll somit eine Methode entworfen werden, die dem Anwender eine zeitlich versetzte Erstellung von Anwenderbeiträgen ermöglicht. Dabei soll ein Annotationssystem zum Einsatz kommen, das die Erstellung von Anwenderbeiträgen erleichtert und die Qualität der Beiträge erhöht.

Im Folgenden werden die Ziele und Lösungskonzepte von OpenProposal erarbeitet, mit denen diese Kriterien erfüllt werden sollen. Zunächst werden Interviews mit Unternehmensvertretern geführt, um die gewonnenen Erkenntnissen aus der Literatur (vgl. Kapitel 3 und 4) an den Erwartungen der Praxis zu spiegeln, Ziele von OpenProposal abzuleiten und darauf aufbauend Lösungskonzepte für das Annotationssystem von OpenProposal zu entwerfen.

6.2.1 Ziel Z1: Vereinfachung der Erstellung von Anwenderbeiträgen

Um die Belastung der Anwender zu reduzieren, sollte die Erstellung von Anwenderbeiträgen mit OpenProposal so einfach und schnell wie möglich bzw. notwendig erfolgen, um eine hohe Akzeptanz beim Anwender sicherzustellen. Dabei sollten den Anwendern auch unnötige Aktivitäten erlassen werden, damit sie sich so weit wie möglich auf ihre Anwenderbeiträge konzentrieren können.

Vom Anwender wird erwartet, dass er seine Sicht auf die zu entwickelnde Software in das Projekt einbringt, um später besser mit einer an seine Bedürfnisse angepassten Software arbeiten zu können. Dabei ist zu beachten, dass der Anwender in den meisten Fällen in Vollzeit mit seiner eigentlichen Arbeit beschäftigt ist und sich für die Verbesserung von Software-Projekten zusätzliche Zeit nehmen muss. Daher ist eine effiziente und effektive Beteiligung anzustreben, so dass der Anwender das Gefühl bekommt, seine Zeit sinnvoll zu investieren. In erster Linie erfordert dies die Möglichkeit einer schnellen Abgabe von Anwenderbeiträgen ohne aufwändige Einarbeitungszeit und mit geringem Aufwand bei der Erstellung. Zudem sollte dem Anwender nur eine geringe kognitive Belastung bei der Bedienung des Annotationssystems zugemutet werden, damit er sich soweit wie möglich auf die kreative Tätigkeit konzentrieren kann.

Das Erstellen und Einreichen von Anwenderbeiträgen sollte somit in wenigen Schritten vollständig elektronisch und medienbruchfrei erfolgen können. Die Einarbeitungszeit des Anwenders sollte nur wenige Minuten betragen, die Bedienung des Annotationssystems daher den üblichen Standards zur Gebrauchstauglichkeit genügen und sich schließlich an den gängigen Bedienkonzepten orientieren.

Für das Annotationssystem von OpenProposal wurde eine einfache und herkömmliche Benutzerinteraktion gewählt, die Anwendern aus anderen Software-Produkten bekannt sein sollte. Abbildung 6.1 zeigt eine schematische Darstellung des Annotationssystems. Es besteht aus einer Werkzeugleiste (1) mit Schaltflächen (2), deren Auswahl die Funktionalitäten auslöst. Das Werkzeug lässt sich über Standardschaltflächen (3) minimieren und beenden.

Abbildung 6.1: Konzept der Bedienoberfläche von OpenProposal

Hinsichtlich der Qualität der Anwenderbeiträge sollten diese leicht verständlich und eindeutig interpretierbar sein, so dass kein weiterer Aufwand für Rückfragen notwendig wird und auch keine Missverständnisse auftreten können. Studien (vgl. [Grudin, 1991a; Holtzblatt und Beyer, 1995; Iivari und Iivari, 2006; Kujala, 2008] zeigen, dass es den Anwendern zu Projektbeginn häufig schwer fällt, konkrete Angaben zu machen, da sie sich zu diesem Zeitpunkt oft nicht vorstellen können, was möglich ist und was sinnvoll sein kann. Je konkreter die Möglichkeiten und Alternativen dargestellt werden (z.B. mithilfe von Prototypen, Layout-Entwürfen, Lastenheft, Prozessmodellen, etc.), desto konkreter und verständlicher kann der Anwender seine Wünsche formulieren.

Das Annotationssystem von OpenProposal baut daher auf der Idee auf, dass der Anwender das Bildschirmfoto („snapshot“) eines Prototypen während der Benutzung des Prototypen annotiert. Das Bildschirmfoto dient damit als zentrales Artefakt eines Anwenderbeitrags. Der Anwender kann seinen Beitrag mit Hilfe des Annotationssystems auf der grafischen Benutzeroberfläche der getesteten Software bzw. des Prototypen platzieren, und bei Bedarf eine textliche Beschreibung hinzufügen. Mit der Annotation kann der Anwender markieren, auf welches Element der Software er sich bezieht. Abbildung 6.2 beschreibt exemplarisch das Konzept der Vorschlagserstellung mit OpenProposal. Der Anwender startet OpenProposal im Hintergrund und beginnt mit dem Test der Software. Fällt dem Anwender eine Verbesserungsmöglichkeit auf, aktiviert er OpenProposal. Beim Aktivieren (2) wird ein Bildschirmfoto des gesamten Anwendungsbildschirms erstellt. Dieses Bildschirmfoto wird auf dem Anwendungsbildschirm eingeblendet, so dass dem Anwender suggeriert wird, dass die Anwendung „eingefroren“ wurde. Mit den Annotationswerkzeugen (3) kann der Anwender anschließend Annotationen platzieren. Hierzu markiert er die zu verbessernde Stelle der Software und fügt eine textliche Beschreibung hinzu.

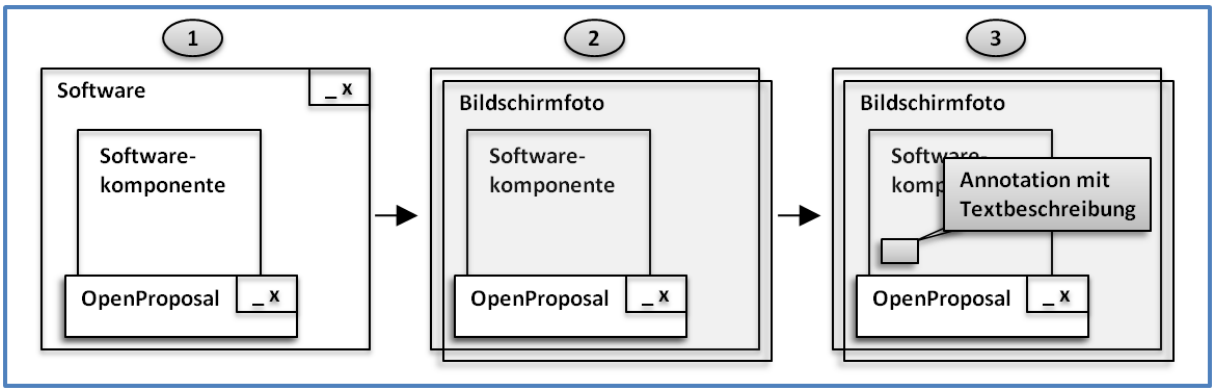

Abbildung 6.2: Konzept der Vorschlagserstellung mit OpenProposal

Im Annotationssystem von OpenProposal sind für diesen Schritt die drei Schaltflächen „Aktivieren/ Deaktivieren“, „Annotieren“ und „Absenden“ notwendig, wie in Abbildung 6.3 dargestellt. Mit „Aktivieren/ Deaktivieren“ wird das Bildschirmfoto aktiviert und dem Anwender somit das Annotieren ermöglicht. Um das Annotieren zu pausieren bzw. um wieder auf die Software zugreifen zu können, kann der Anwender das Annotationssystem durch ein erneutes Drücken auf die Schaltfläche deaktivieren. Die Software wird freigegeben und der Anwender kann beispielsweise die Software weiter testen oder auch die Fensteranordnung anpassen. Über die Schaltfläche „Annotieren“ kann der Anwender die Annotationsobjekte auf das Bildschirmfoto setzen. Mit „Absenden“ wird das Bildschirmfoto gemeinsam mit den Annotationen an den Moderator versendet. Ferner kann ein bereits abgesendeter bzw. gespeicherter Anwenderbeitrag mithilfe der Schaltfläche „Vorschlag öffnen“ geöffnet, überarbeitet und mittels „Absenden“ wieder gespeichert werden.

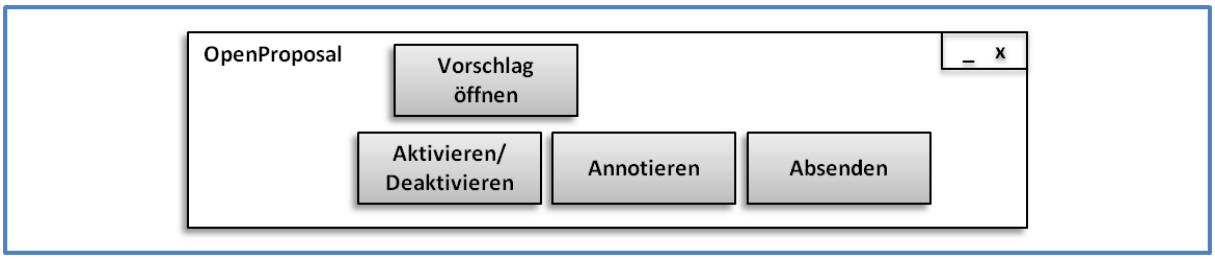

Abbildung 6.3: Konzept der Hauptbedienelemente des Annotationssystems

Zur Vereinfachung sollen dem Anwender durch eine teilweise Automatisierung der Dateneingabe unnötige Arbeitsschritte bei der Erstellung von Anwenderbeiträgen erspart werden. So werden der Name und weitere Programmparameter (z.B. Versionsnummer, Parametrisierung, aktive Module, etc.) der annotierten Software aus der Systemumgebung entnommen. Aus der Annotation und den Systemparametern werden der Titel und eine Kurzbeschreibung zum Anwenderbeitrag generiert. Die Einstellungen zum Absenden können vorab konfiguriert werden, so dass eingestellt werden kann, wie der Vorschlag gespeichert werden soll. Somit braucht sich der Anwender nicht damit auseinanderzusetzen, in welcher Form er seinen Anwenderbeitrag an den Moderator absendet.

6.2.2 Ziel Z2: Formale Strukturen zur besseren Verständlichkeit

Da Entwickler klare und nachvollziehbare Erklärungen vorziehen, soll Anwendern eine formale Struktur zur Formulierung ihrer Anwenderbeiträge angeboten werden. Dies soll zu leichter verständlichen und konkreteren Anwenderbeiträgen führen und Kommunikationskosten senken. Hierzu gehört

auch, dass der Anwender sich auf konkrete Bereiche der Software bezieht und seine Meinung bzw. Wünsche einer Kategorie zuordnet. Zudem soll über die Speicherung von Bildschirmfoto, Systemparametern und evtl. Fehlercodes der Nutzungskontext erfasst werden. Über das Annotationssystem soll dem Anwender daher eine Assistenz angeboten werden, die ihn beim Annotieren unterstützt und ihm formale Strukturen gibt, um alle notwendigen Informationen in einer verständlichen Weise zu erhalten.

6.2.3 Ziel Z3: Vereinfachung der Vor- und Nachbereitung

Zur Vereinfachung bzw. Effizienzsteigerung der Vor- und Nachbereitung der Anwenderbeteiligung sollen die dafür erforderlichen Arbeitsschritte mit OpenProposal auf das Wesentliche reduziert werden. Außerdem sollte die Nutzung des Annotationssystems von OpenProposal so wenige Voraussetzungen wie möglich an Menschen und Infrastruktur stellen und sich an bereits vorhandene Rahmenbedingungen anpassen lassen. Dies erfordert auch, dass die Anwenderbeteiligung mit OpenProposal sowohl im Rahmen von Testinstallationen in einem Testlabor als auch im Nutzungskontext in der alltäglichen Arbeitsumgebung möglich ist.

Abgegebene Anwenderbeiträge sollen in einer kollaborativen webbasierten Umgebung gespeichert werden, die allen Teilnehmern einen bedarfsgerechten Zugang zu den abgegebenen Vorschlägen erlaubt und als Kommunikationsplattform für das Anforderungsmanagement und die Entwicklungsteams dient. Da die Anwenderbeiträge mitunter nachträglich modifiziert werden müssen (z.B. bei unvollständigen Informationen, unklaren Vorschlägen, etc.), sollte die Möglichkeit gegeben sein, die Anwenderbeiträge auch nach deren Abgabe zu bearbeiten. Zudem sollte es möglich sein, jeden Anwenderbeitrag als einzeln identifizierbares Dokument in übersichtlicher und schnell erfassbarer Form zu erhalten und dabei auch die Häufigkeit ähnlicher bzw. zusammenhängender Anwenderbeiträge überblicken zu können. Mithilfe der webbasierten Kollaborationsumgebung sollen auch die Entscheidungsprozesse beschleunigt werden, indem der Moderator die Anwenderbeiträge nach Eingang in der Plattform auf ihre Eignung prüft, ggf. anpasst und anschließend für den Entscheider aufbereitet.

Das Speichern des Nutzungskontexts in Form des Bildschirmfotos und der Systemparameter bzw. Fehlercodes soll darüber hinaus automatisch im Hintergrund erfolgen, ohne dass sich der Anwender bewusst darum kümmern muss.

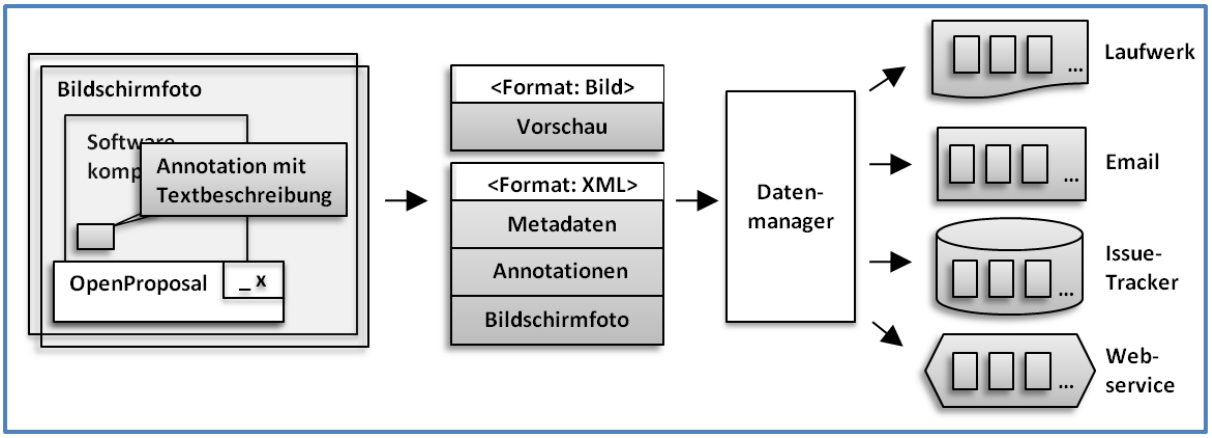

Abbildung 6.4: Konzept der Speicherung der Anwenderbeiträge im Annotationssystem

Das Lösungskonzept zur Speicherung und Verwaltung von Anwenderbeiträgen im Annotationssystem von OpenProposal illustriert Abbildung 6.4. Nach Drücken der Schaltfläche „Absenden“ wird der Vorschlag in zwei Dokumenten gespeichert. Im ersten Dokument wird der Anwenderbeitrag in Form eines Bilddokuments als Vorschau generiert und enthält das Bildschirmfoto mit den darauf platzierten Annotationen. Die Vorschau wird als Bilddatei gespeichert. Das zweite Dokument ist im XML-Format gehalten, besteht aus drei Teilbereichen und beinhaltet den vollständigen Datensatz des Anwenderbeitrags. Der erste Bereich beschreibt die Metadaten zum Vorschlag, wie z.B. Titel, Kurzbeschreibung, Anwendungskontext, etc. Im zweiten Bereich werden die Objektinformationen der Annotationen wie z.B. Kategorie, Position oder Textbeschreibung gespeichert. Der dritte Bereich enthält das Original-Bildschirmfoto ohne die Annotationen. Die Metadaten, die Annotationen und das Bildschirmfoto werden in einer XML-Datei gespeichert. Die Anzeige eines Anwenderbeitrags ist somit auf zwei Wegen möglich:

- Die Bilddatei zeigt das Bildschirmfoto mit den Annotationen. Dieses Bild kann mit jedem Bildbetrachtungsprogramm dargestellt werden. In dieser Darstellung kann nur das annotierte Bildschirmfoto ohne die Daten zum Nutzungskontext betrachtet werden. Dies ist hilfreich, um einen schnellen Überblick über die Anwenderbeiträge zu erhalten.

- Die XML Datei kann mit dem Annotationssystem von OpenProposal geöffnet werden. Es zeigt das Bildschirmfoto im Hintergrund an und positioniert die Annotationen gemäß den Objektdaten auf das Bildschirmfoto. Die Metadaten werden in einem Textfeld angezeigt. Mit dieser Darstellung ist eine vollständige Darstellung der Anwenderbeiträge sowie eine weitere Anpassung der Anwenderbeiträge möglich.

Zur Speicherung der Anwenderbeiträge sollen unterschiedliche Speicherorte, z.B. Web-Services, Issue-Tracker, Email oder ein Laufwerk, angesteuert werden können. Eine direkte Übermittlung in einen Issue-Tracker sollte möglich sein, um den Anwenderbeitrag direkt in die Kollaborationsplattform bzw. Entwicklungsumgebung der Entscheider und Entwickler zu übertragen. Dies erspart dem Moderator den Aufwand, die Anwenderbeiträge manuell zu transferieren. Allerdings kann nicht immer davon ausgegangen werden, dass ein Issue-Tracker direkt zugänglich bzw. überhaupt vorhanden ist, etwa dann, wenn zu Hause oder in einem Testlabor kein Netzwerkzugang installiert ist. Außerdem besteht die Möglichkeit, dass der Zugang zum Issue-Tracker nur bestimmten Mitarbeitern vorbehalten ist. In solchen Situationen ist eine flexible Anbindungsmöglichkeit von Vorteil, um unabhängig von den Rahmenbedingungen handeln zu können. Neben dem Issue-Tracker bietet sich hier die Alternative an, die Vorschläge auf einem Laufwerk bzw. Netzlaufwerk abzulegen, als Email zu versenden oder an einen Webservice zu übermitteln. Die Einstellungen zur Speicherung der Anwenderbeiträge werden im Datenmanager vorgenommen.

Im Issue-Tracker wird der Anwenderbeitrag als neue Aufgabe bzw. „Issue“ eingetragen. Absender, Titel und Kurzbeschreibung des Issue werden wie bei der Email aus den Metadaten und den Annotationen generiert. An dieses Issue werden die Bilddatei und die XML-Datei als Anhang angefügt. Über den Issue-Tracker kann der Moderator das annotierte Bildschirmfoto zu jedem Issue als Bilddatei betrachten oder als XML-Dokument im Annotationssystem öffnen.

Eine Alternative stellt die Speicherung des Anwenderbeitrags als Datei auf einem Laufwerk dar. Dies ist die flexibelste Möglichkeit, da hierzu nur ein Laufwerk als Speicherort angegeben werden muss

und keine Netzwerkverbindung notwendig ist. Allerdings müsste der Moderator die Dateien vom Laufwerk abrufen können oder auf andere Weise vom Anwender zur Verfügung gestellt bekommen, um diese weiter bearbeiten zu können. Für den Moderator kann dabei zusätzlicher Aufwand für das Einstellen der Vorschläge in den Issue-Tracker entstehen. Hier kann das Annotationssystem dazu beitragen, den Aufwand zu reduzieren, indem die Anwenderbeiträge im Annotationssystem geöffnet werden und über das Annotationssystem direkt im Issue-Tracker abgespeichert werden (siehe hierzu auch die nachfolgenden Erläuterungen zu Abbildung 6.6).

In einer Email-Variante wird der Anwenderbeitrag als Email mit Anhang versendet. Der Absendername, Betreff und die Nachricht der Email werden aus den Metadaten und den Annotationen generiert. Im Anhang der Email werden die Bilddatei und die XML-Datei angehängt. Hierzu sind Einstellungen zum Email-Versand wie z.B. Email-Empfänger, Email-Postausgangsserver, etc. vorzunehmen. Der Moderator kann sich die Anwenderbeiträge somit direkt als Email senden lassen. Allerdings setzt diese Variante eine Netzwerkverbindung, Email-Konten und einen Email-Postausgangsserver voraus. Auch hier müssen die Anwenderbeiträge für die weitere Verarbeitung in einen Issue-Tracker eingestellt werden. Moderne Issue-Tracking-Systeme wie z.B. Atlassian JIRA[8] erlauben die Erstellung von Issues per Email, sodass der Moderator die per Email eingetroffenen Anwenderbeiträge als Email an den Issue-Tracker weiterleiten kann.

Mit der Anbindung von Webservices kann die Abwicklung der Speicherung der Anwenderbeiträge vollständig extern, also unabhängig vom Annotationssystem, gesteuert werden. Dafür muss das Annotationssystem eine Schnittstelle enthalten, um den Zugriff auf Webservices zu regeln. Ein Webservice benötigt eine Schnittstelle zum Empfang der XML-Datei sowie der Bilddatei. Diese Schnittstelle wird über das Annotationssystem aufgerufen, um die Dateien zu übertragen. Die weitere Verfahrensweise kann im Webservice implementiert werden. Dieser Ansatz bietet die Möglichkeit, den Prozess der Erstellung der Anwenderbeiträge an die vorhandene Infrastruktur anzupassen, ohne in die Implementierung des Annotationssystems einzugreifen. Hierbei ist mit einem erhöhten Aufwand bei der Anpassung von Webservices und der Einrichtung der dafür notwendigen Infrastruktur zu rechnen.

Mit dem Datenmanager können zudem Regeln definiert werden, die eine verfeinerte Einstellung der Speicherung der Anwenderbeiträge erlauben. Diese Regeln basieren auf einer „Wenn“-Kondition und einer „Dann“-Maßnahme. Die „Wenn“-Kondition kann auf die Daten im Anwenderbeitrag bezogen werden. Die „Dann“-Maßnahme kann die Parameter bzw. Variablen zum Speichern eines Anwenderbeitrages verändern.

Folgendes Szenario ist beispielsweise denkbar: Der Anwender testet eine Internetanwendung und benutzt das OpenProposal Annotationssystem, um seine Anwenderbeiträge zu erstellen. Er findet einen Schreibfehler auf der Startseite, annotiert den Fehler und sendet die Fehlermeldung ab. Über das Auslesen der Systemparameter werden die aktiven Anwendungen identifiziert, u.a. auch der Internetbrowser. Der Abgleich der Position der Annotation mit der Fensterposition des Internet-

[8] Atlassian JIRA, http://www.atlassian.com/software/jira/, abgerufen am 07.03.2009

browsers sowie die Information, dass das Fenster des Internetbrowsers zum Zeitpunkt der Feedbackabgabe fokussiert und aktiv war, erlaubt die Schlussfolgerung, dass sich die Annotation auf die geöffnete Internetseite im Internetbrowser bezieht. Die URL-Adresse gibt Auskunft darüber, um welche Internetseite es sich handelt.

Mit Hilfe dieser Daten kann der Datenmanager so konfiguriert werden, dass Verbesserungsvorschläge zur Startseite der Internetanwendung im Issue-Tracker in die Kategorie des Projektes und in die Unterkategorie „Startseite“ eingetragen werden. Die Definition einer solchen Regel könnte in einer Form entsprechend Abbildung 6.5 dargestellt werden.

```
Wenn
        $AnnotierteSoftwareName = “Internetbrowser” UND
        $AnnotierteSoftwareURL = “vorhanden“ UND
        $AnotierteSoftwareURL = “Startseite”
Dann
        $IssueTrackerProjekt = “Internetanwendung”;
        $IssueTrackerKategorie = “Startseite”;
```

Abbildung 6.5: Pseudocode für eine beispielhafte Regeldefinition im Datenmanager

Das Konzept zur Nachbereitung der OpenProposal-Vorschläge ist in Abbildung 6.6 skizziert. Unabhängig vom Speicherort kann die XML-Datei mit OpenProposal geöffnet und überarbeitet werden. Die überarbeitete Version wird wie ein neuer Vorschlag behandelt und über den Datenmanager gespeichert. Dem Moderator steht es anschließend frei, ob er die vorherige Version weiterhin speichert oder löscht. Eine Möglichkeit zur Versionisierung soll nicht angeboten werden, um die Komplexität der Benutzerinteraktionen mit dem Annotationssystem möglichst niedrig zu halten.

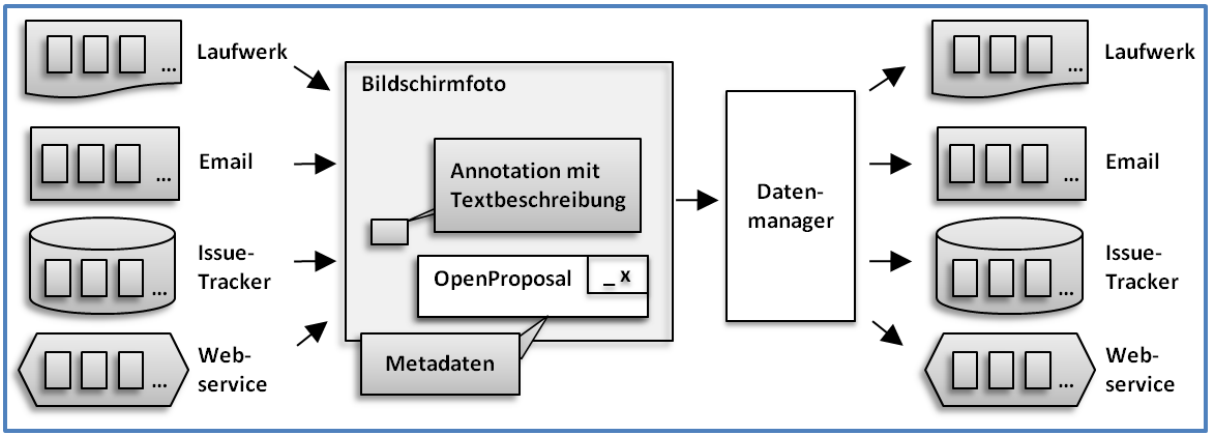

Abbildung 6.6: Konzept der Nachbereitung der Anwenderbeiträge im Annotationssystem

6.3 Anforderungen an das Annotationssystem

Aufbauend auf den Zielen und Lösungskonzepten werden im Folgenden die Anforderungen an das Annotationssystem definiert, die im Rahmen der OpenProposal Methode zum Einsatz kommen sollen. Dabei durchläuft die Anforderungsanalyse mehrere Iterationen. Die erste Iteration besteht aus einer Literaturanalyse, welche in Anforderungen an das Annotationssystem aus den unterschiedlichen Perspektiven der beteiligten Rollen resultiert. Anschließend werden Projektdaten aus Projekten

analysiert, um die Anforderungen der Literaturanalyse zu prüfen und weitere Rückschlüsse auf Anforderungen an das Annotationssystem zu ziehen. Schließlich werden mithilfe der identifizierten Anforderungen die Ziele von OpenProposal auf Konsistenz und Vollständigkeit geprüft und bei Bedarf ergänzt bzw. überarbeitet.

6.3.1 Anforderungen aus der Literatur

Um aus erarbeiteten Zielen und Lösungskonzepten konkrete Anforderungen an das Annotationssystem zu formulieren, sind der zu verbessernde Prozess aus Sicht der Beteiligten zu analysieren und deren Anforderungen zu erfassen. In diesem Unterkapitel wird aufbauend auf bisherigen Forschungsarbeiten, etablierten Standards und Richtlinien zur Anwenderbeteiligung (vgl. Kapitel 2.4 und 2.7) eine initiale Liste von Anforderungen erstellt.

Zu Beginn werden allgemeine Anforderungen erhoben, die alle Beteiligten gleichermaßen betreffen. Anschließend werden aus Sichtweise der jeweiligen Prozessbeteiligten mögliche konkrete Wünsche diskutiert und Anforderungen formuliert. Der Fokus der Untersuchungen liegt aufgrund der identifizierten Ziele auf den Prozessen zur Erstellung von Anwendervorschlägen und der Vor- und Nachbereitung einer Anwenderbeteiligung.

6.3.1.1 Allgemeine Anforderungen

Bei der Entwicklung eines Annotationssystems ist im Allgemeinen die Gebrauchstauglichkeit sicherzustellen, deren Kriterien in den Richtlinien und Empfehlungen der ISO Normreihe 9241 (EN ISO 9241) und der ISO/IEC 9126 (vgl. Kapitel 2.7.1) festgehalten sind. Anforderung A1 lautet somit:

A1: „Gebrauchstauglichkeit“ - Die Gebrauchstauglichkeit nach den Richtlinien der ISO Normreihe 9241 (EN ISO 9241) und der ISO/IEC 9126 ist sicherzustellen.

In Bezug auf die Anwendbarkeit eines Annotationssystems ist zu empfehlen, dass das Annotationssystem soweit wie möglich unabhängig von einer bestimmten Plattform eingesetzt werden kann. So kann dem Umstand Rechnung getragen werden, dass bei einer frühen Anwenderbeteiligung erste Entwürfe in Form von Grafiken, später ein erster lauffähiger Prototyp und zum Ende hin eine prefinale Version der Software getestet wird. Eine Beschränkung auf eine bestimmte Kategorie von Software bzw. Plattform würde die Möglichkeiten zum Einsatz des Annotationssystems reduzieren. Dies führt zur ersten Anforderung A2 zum plattformunabhängigen Einsatz eines Annotationssystems:

A2: „Plattformunabhängigkeit“ - Die Anwenderbeiträge sollten unabhängig von der Art der Software erstellt werden können.

Auch wenn Papier im Vergleich zu elektronischen Medien unübertreffbare Vorteile in seiner Haptik und Bedienfreundlichkeit aufweist, verursacht es als Kommunikationsmedium Medienbrüche durch einen Wechsel des Kommunikationsmediums innerhalb einer Informationsverarbeitungsprozesses. Medienbrüche führen zu einer drastischen Reduktion der Produktivität kollaborativer Aktivitäten. In der Literatur (vgl. [Geißler et al., 2004; Keil-Slawik und Selke, 1998; Mikkelsen und Aasly, 2001] herrscht hierzu in unterschiedlichen Branchen und Forschungsrichtungen einhellig die Meinung, dass eine kollaborative Tätigkeit medienbruchfrei und somit ausschließlich elektronisch erfolgen sollte.

Das „Abtippen“ von papierbasierten Dokumenten und der damit verbundene Aufwand und Zeitverlust entfallen. Somit lautet Anforderung A3:

A3: „Elektronische Datenhaltung“ - Anwenderbeiträge sollten durchgängig in elektronischer Form gespeichert werden.

6.3.1.2 Anforderungen aus Sicht des Entwicklers

Der Entwickler befindet sich in der Situation, die Anwenderbeiträge in kurzer Zeit nachzuvollziehen und korrekt umzusetzen (vgl. Kapitel 2.4.1, 2.7 und 2.9). Dem Entwickler muss daher die Möglichkeit gegeben werden, sich in kurzer Zeit einen Überblick über alle relevanten Informationen zu den Anwenderbeiträgen zu verschaffen. Anforderung A4 lautet somit:

A4: „Leichte bzw. schnelle Verständlichkeit der Anwenderbeiträge“ - Der Entwickler kann die Anwenderbeiträge mit einem niedrigen Aufwand nachvollziehen.

Nach den Bewertungskriterien für Methoden der Anwenderbeteiligung (vgl. Kapitel 2.9) und dem Verständnis für Gebrauchstauglichkeit (vgl. Kapitel 2.7.1) erweist sich neben der reinen Vorschlagsbeschreibung auch der Nutzungskontext, in dem sich der Anwender bei der Vorschlagsabgabe befindet, für die Entwickler als interessant. Daher lautet Anforderung A5:

A5: „Erfassung des Nutzungskontextes des Anwenders“ - Die Anwenderbeiträge enthalten Informationen über die Situation des Anwenders während der Erstellung der Anwenderbeiträge und helfen dem Entwickler dabei, den Nutzungskontext des Anwenders nachzuvollziehen.

Die Kommunikation der durchgeführten Anpassungen am Software-Produkt stellt einen wichtigen Akzeptanzfaktor sowohl bei Anwendern als auch bei Entwicklern dar, wie er auch bei den Methoden zur Anwenderbeteiligung zwingend vorausgesetzt wird (vgl. Kapitel 2.7.4). Es ist demzufolge zu empfehlen, dass Entwickler und Entscheider den Anwendern Einblicke in die Hintergründe der Entscheidungsfindung und in die notwendigen Aufwände ermöglicht, um so zu garantieren, dass die durchgeführte Arbeit auch entsprechend wertgeschätzt wird und der Anwender sich an der Entwicklung beteiligt fühlt. Anforderung A6 lautet somit:

A6: „Transparenz des Bearbeitungsstatus“ - Der Entwickler kann den Bearbeitungsstatus der Anwenderbeiträge kommunizieren.

Für die Produktivität der Entwickler ist es förderlich, wenn die bestehende Entwicklungsumgebung der Entwickler so weit wie möglich genutzt werden kann. Whitehead [2007] weist darauf hin, dass die Anwenderbeiträge aus Sicht des Entwicklers im Idealfall in ein Issue-Tracking-System einfließen und mit dem Anforderungsmanagement abgeglichen werden sollten. Zudem bieten die gängigen Entwicklungsumgebungen (z.B. Eclipse, JBuilder, Visual Studio) die Integration von Issue-Trackern in die Benutzeroberfläche der Entwicklungsumgebung an, um so Aufgabenmanagement und Entwicklungsumgebung der Entwickler auf einer Oberfläche anzubieten. Anforderung A7 lautet somit:

A7: „Integration der Entwicklungsumgebung“ - Die Anwenderbeiträge können in vorhandene Entwicklungsumgebungen integriert werden.

6.3.1.3 Anforderungen aus Sicht des Anwenders

Für den Anwender ist eine effiziente und effektive Beteiligung wichtig, um das Gefühl zu haben, dass seine Zeit sinnvoll investiert wird (vgl. Kapitel 2.7.4). In erster Linie erfordert dies die Möglichkeit einer schnellen Erstellung von Anwenderbeiträgen ohne aufwändige Einarbeitungszeit und mit niedrigem Aufwand. Zudem sollte dem Anwender nur eine geringe kognitive Belastung bei der Bedienung des Annotationssystems zugemutet werden. Anforderung A8 lautet somit:

A8: „Effizienz der Erstellung von Anwenderbeiträgen“ - Der Anwender kann seine Anwenderbeiträge mit niedrigem Aufwand und Einarbeitungszeit abgeben.

Der Einsatz von Prototypen ist für eine effektive Anwenderbeteiligung (vgl. Kapitel 2.7.4) wichtig. Da Prototypen in unterschiedlicher Form (z.B. lauffähige Software, Zeichnungen, Mock-Ups) auftreten, sollte das Annotationssystem auf unterschiedliche Plattformen und Repräsentationen angewendet werden können. Dieser Anspruch würde die bereits aufgestellte Anforderung A2 folgendermaßen erweitern:

A2: „Plattformunabhängigkeit“ - Die Anwenderbeiträge sollten unabhängig von der Art und Repräsentation der Software erstellt werden können.

Zudem sollte der Fall berücksichtigt werden, dass dem Anwender nach Abgabe seiner Anwenderbeiträge Ergänzungen einfallen. Daraus ergibt sich Anforderung A9:

A9: „Änderbarkeit der Anwenderbeiträge“ - Der Anwender kann seine Anwenderbeiträge nachträglich ändern.

Die Anwender möchten die Resultate ihrer Beiträge einsehen und nachvollziehen können (vgl. Kapitel 2.7.4). Dies stellt einen wichtigen Motivationsfaktor der Anwender zur Erstellung von Anwenderbeiträgen. Dieser Forderung kann in Anforderung A4 „Transparenz des Bearbeitungsstatus“ mit einer Ergänzung entsprochen werden:

A4: „Transparenz des Bearbeitungsstatus“ - Der Entwickler kann den Bearbeitungsstatus der Anwenderbeiträge kommunizieren. Der Anwender kann sich über den Bearbeitungsstand seiner Anwenderbeiträge informieren.

Zudem spielt die Garantie der Vertraulichkeit eine wichtige Rolle, um das Vertrauen der Anwender zu gewinnen und die Ängste vor einer Überwachung zu nehmen. Bei dem Test von Software fühlen sich Anwender erwiesenermaßen häufig einem Prüfungsstress bzw. Erfolgsdruck ausgesetzt (vgl. [Nielsen, 1994a; Rubin et al., 2008; Sarodnick und Brau, 2006]). Im Annotationssystem sollte es die Möglichkeit geben, nur solche Informationen zu erfassen, die für den Test der Software relevant sind und nicht zum Nachteil des Anwenders verwendet werden können. Anforderung A10 lautet somit:

A10: „Vertraulichkeit“ - Der Anwender kann seine Anwenderbeiträge auf Wunsch anonymisiert abgeben. Rückschlussmöglichkeiten auf seine Person und auf private Daten werden so weit wie möglich verhindert.

Weiterhin ist zu beachten, dass der Anwender nicht nur in einer Befragungssituation sondern auch während des Arbeitsalltags Verbesserungsideen entwickelt (vgl. Kapitel 2.7.4). Anforderung A11 lautet somit:

A11: „Erstellung von Anwenderbeiträgen im Nutzungskontext“ - Der Anwender kann seine Anwenderbeiträge während der alltäglichen Nutzung der Software abgeben.

6.3.1.4 Anforderungen aus Sicht des Moderators

Bei der Identifizierung der Anwenderbedürfnisse ist es schwierig, abzuschätzen, wie wichtig und dringend die angebrachten Bedürfnisse und Verbesserungsvorschläge sind. Ein Vorschlag kann für eine einzelne Person als dringend, aber für seine Abteilung insgesamt als unwichtig gelten, während er wiederum für weitere Mitarbeiter aus anderen Abteilungen von hoher Bedeutung sein könnte. Daher ist eine organisationsübergreifende Übersicht über die Bedürfnisse der Anwender eine zentrale Aufgabenstellung für den Moderator (vgl. Kapitel 2.4.4). Dies führt zu Anforderung A12:

A12: „Übersichtliche Darstellung und Anpassung der Anwenderbeiträge“ - Der Moderator kann die Anwenderbeiträge der Anwender verwalten und anpassen.

6.3.1.5 Anforderungen aus Sicht des Entscheiders

Der Entscheider ist in erster Linie für die Zustimmung bzw. Ablehnung von Verbesserungsvorschlägen, die Vergabe von Prioritäten der Anforderungen und das Anforderungsmanagement zuständig. Wird einer Anforderung zugestimmt, ist die Anforderung im Anforderungsmanagement zu erfassen und zu verwalten (vgl. Kapitel 2.3 und 2.4.3). Für das Anforderungsmanagement existieren bereits Werkzeuge, die im Regelfall auch in der Entwicklungsumgebung der Entwickler integriert sind, so dass für diesen Fall die Anforderung A5 „Integration der Entwicklungsumgebung“ Gültigkeit besitzt.

Zudem sollte aus Sicht des Entscheiders die Zeit für die Entscheidungsfindung und für den Kommunikationsprozess zwischen Anforderungsanalyse, Entscheidung und Umsetzung kurz gehalten werden, um Entscheidungen in einem Projekt zeitnah treffen und bald mit den Umsetzungen beginnen zu können (vgl. Kapitel 2.4.3). Daraus ergibt sich Anforderung A13:

A13: „Zeitnahe Entscheidungsfindung“ - Der Entscheider erhält die Anwenderbeiträge zeitnah nach ihrer Erstellung.

6.3.1.6 Zusammenfassung der Anforderungen aus der Literatur

Zusammenfassend können aus den allgemeinen Anforderungen sowie den einzelnen Anforderungen der beteiligten Gruppen zahlreiche konkrete Anforderungen an das Annotationssystem von OpenProposal formuliert werden, wie in Tabelle 6.1 aufgelistet.

Nr.	Anforderung
A1	„Gebrauchstauglichkeit“ - Die Gebrauchstauglichkeit nach den Richtlinien der ISO Normreihe 9241 (EN ISO 9241) und der ISO/IEC 9126 ist sicherzustellen.
A2	„Plattformunabhängigkeit“ - Die Anwenderbeiträge sollten unabhängig von der Art und Darstellungsformat der Software formuliert werden können. Das Darstellungsformat nimmt die Form einer lauffähigen Software oder eines Dokumentes zu einer Software ein.
A3	„Elektronische Datenhaltung“ - Anwenderbeiträge sollten durchgängig in elektronischer Form gespeichert werden.
A4	„Leichte bzw. schnelle Verständlichkeit der Anwenderbeiträge“ - Der Entwickler kann die Anwenderbeiträge mit einem niedrigen Aufwand nachvollziehen.
A5	„Erfassung des Nutzungskontextes des Anwenders“ - Die Anwenderbeiträge enthalten Informationen über die Situation des Anwenders während der Erstellung der Anwenderbeiträge und helfen dem Entwickler dabei den Nutzungskontext des Anwenders nachzuvollziehen.
A6	„Transparenz des Bearbeitungsstatus“ - Der Entwickler kann den Bearbeitungsstatus der Anwenderbeiträge kommunizieren. Der Anwender kann sich über den Bearbeitungsstand seiner Anwenderbeiträge informieren
A7	„Integration der Entwicklungsumgebung“ - Die Anwenderbeiträge können in vorhandene Entwicklungsumgebungen integriert werden.
A8	„Effizienz der Erstellung von Anwenderbeiträgen“ - Der Anwender kann seine Anwenderbeiträge mit niedrigem Aufwand und Einarbeitungszeit abgeben.
A9	„Änderbarkeit der Anwenderbeiträge“ - Der Anwender kann seine Anwenderbeiträge nachträglich ändern.
A10	„Vertraulichkeit“ - Der Anwender kann seine Anwenderbeiträge auf Wunsch anonymisiert abgeben. Rückschlussmöglichkeiten auf seine Person und auf private Daten werden so weit wie möglich verhindert.
A11	„Erstellung von Anwenderbeiträgen im Nutzungskontext“ - Der Anwender kann seine Anwenderbeiträge während der alltäglichen Nutzung der Software abgeben.
A12	„Übersichtliche Darstellung und Anpassung der Anwenderbeiträge“ - Der Moderator kann die Anwenderbeiträge der Anwender verwalten und anpassen.
A13	„Zeitnahe Entscheidungsfindung“ - Der Entscheider erhält die Anwenderbeiträge zeitnah nach dessen Abgabe.

Tabelle 6.1: Anforderungen aus der Literatur an das Annotationssystem von OpenProposal

6.3.2 Analyse von Projektdaten

Im Rahmen von Software-Projekten werden häufig Email und Issue-Tracker eingesetzt, um von den Anwendern Rückmeldungen zu den aktuellen Entwicklungen zu erhalten. Da die Daten elektronisch gehalten und oft auch archiviert werden, stellen sie eine gute Quelle zur Analyse des Kommunikationsverhaltens zwischen Anwendern, Moderatoren, Entscheidern und Entwicklern dar. Auch wenn ein großer Teil der Kommunikation über andere Kommunikationskanäle verläuft, kann die Analyse von Konversationen via Email und Issue-Tracker ergänzende Hinweise bieten.

Anlehnend an die Methodik der qualitativen Inhaltsanalyse nach Mayring [2007] wird im Folgenden die Analyse von text- und grafikbasierten Verbesserungsvorschlägen aus Projektdatenbanken durchgeführt. Es wird der Frage nachgegangen, ob Widersprüche zu den bisher aufgestellten Anforderungen existieren und ob Anforderungen hinzugefügt werden sollten. Als Quelle der analysierten Daten dienen Projektdatenbanken aus verschiedenen Software-Projekten. Als Analyseeinheit dient eine Sammlung von Verbesserungsvorschlägen, die als Text- oder Multimediadokumente in Projektdatenbanken vorliegen. Das initiale Anforderungsset in Tabelle 6.1 stellte die Kategorisierung dar, d.h. die bisher erarbeiteten Anforderungen werden als Kategorien herangezogen. Mit Hilfe der Kategorien wird die Analyseeinheit nach übereinstimmenden bzw. sich widersprechenden Inhalten untersucht. Anhand dieser Daten wird die Gültigkeit der aufgestellten Anforderungen überprüft. Finden sich bei der Analyse neue Anforderungen, die nicht von den bis dahin definierten Anforderungen abgedeckt werden, werden sie im Anforderungsset ergänzt. Zur kommunikativen Validierung - d.h. die Ergebnisse werden den beteiligten Personen selbst zur Prüfung vorgelegt - werden die mit der Inhaltsanalyse gewonnenen Erkenntnisse in Interviews mit den verantwortlichen Personen besprochen und abgestimmt.

Es werden Inhalte aus

- einem Issue-Tracker der Abteilung Corporate Finance Center bei Bayer (CoFiPot),
- einer Email-Kommunikation im Software-Projekt Topbraid-Composer (TBC) und
- Anforderungsdokumenten aus Usability Workshops der Firma TRUMPF (TRUMPF)

untersucht. Die Daten wurden innerhalb der Jahre 2006 und 2007 erzeugt und waren Teil von Software-Projekten, bei denen die Entwicklung vollständig neuer Systeme oder die Verbesserung von Prototypen bzw. stabilen Versionen mit Anwendern diskutiert wurden.

Die Präferenz für diese drei Projekte bzw. dieser drei Unternehmen basiert auf ehemaligen bzw. vorhandenen Kooperationsbeziehungen zu den Unternehmen. Ihre Eignung zur Inhaltsanalyse beruht auf der Tatsache, dass die Projektinhalte aus realen Projekten entstammten, einen längeren Zeitraum erfassen und in einem vollständigen originären Zustand vorliegen. Damit sollte der Vielfalt der Anwenderbeteiligung Rechnung getragen und ein Einblick in typische Anwenderbeiträge gegeben werden können.

6.3.2.1 Analyse der Inhalte eines Issue-Trackers im Unternehmensumfeld (CoFiPot)

Seit Januar 2006 wird von der Abteilung Corporate Finance Center des Konzerns Bayer[9] ein webbasiertes Finanzportal mit der Bezeichnung CoFiPot (vgl. [Vo, 2007; Vo und Elsner, 2007]) betrieben, das konzernweit auf fast allen Kontinenten in Anspruch genommen wird (vgl. Abbildung 6.7). Dieses Portal hat das Ziel, den weltweit agierenden Abteilungen von Bayer einen zentralen Einstieg für alle globalen Finanzangelegenheiten zu bieten und sowohl Finanzinformationen in Form von Reports anzubieten als auch Finanzinformationen über Formulare einzuholen.

In dieses Portal ist ein Issue-Tracker integriert, welcher Anwendern die Möglichkeit bietet, Verbesserungsvorschläge und Fehler über ein Feedbackformular abzugeben und auch den Bearbeitungsstatus abgegebener Vorschläge zu verfolgen. Diese Möglichkeit wurde im Analysezeitraum seit der Inbetriebnahme Anfang 2006 bis August 2008 nur intern in der Abteilung Corporate Finance Center angeboten, um diese Art der Kommunikation intern risikofrei zu testen und keine falschen Erwartungen in anderen Abteilungen zu wecken. Feedback von anderen Abteilungen wurde telefonisch oder im persönlichen Gespräch entgegengenommen, notiert und anschließend zur weiteren Bearbeitung im Issue-Tracker eingetragen. Die Entwickler des Portals konnten sich die Issues zuweisen, den Status der Bearbeitung eintragen und die Issues nach Fertigstellung schließen. Basierend auf Aussagen eines Entwicklers wurde davon ausgegangen, dass ein Großteil der Verbesserungsvorschläge und Fehlermeldungen im Issue-Tracker dokumentiert, eine Vielzahl an Änderungsvorschlägen jedoch direkt umgesetzt oder ausschließlich per Gespräch oder Email kommuniziert wurden.

Abbildung 6.7: CoFiPot [Vo, 2007]

[9] Bayer ist ein weltweit agierender Konzern aus über 350 Unternehmen mit Hauptsitz in Leverkusen, Deutschland. Im Juni 2006 beschäftigte Bayer 110.200 Mitarbeiter mit einem Umsatz von 27,3 Milliarden Euro in 2005.

Die Inhaltsanalyse des Issue-Trackers erstreckt sich über einen Datensatz, der zwischen März 2006 und August 2008 im Issue-Tracker eingegeben wurde und 790 Issues mit über 432 Kommentaren und Anhängen (Attachments) umfasst. Die Daten wurden von Anwendern und Entwicklern der Abteilung Corporate Finance Center per Email oder über ein Helpdesk-System eingegeben und in den Jahren 2007 und 2008 auf Anfrage in unregelmäßigen Abständen als Tabelle exportiert und zur Analyse freigegeben. Die Daten beschreiben den Zeitpunkt der Erstellung, den anonymisierten Namen des Feedback-Gebenden, den Titel und die Beschreibung des Sachverhaltes, eine Prioritätszuordnung, eine Typzuordnung, eine Projektzuordnung, eine initiale Aufwandsabschätzung, eine tatsächliche Aufwandserfassung, den anonymisierten Namen des Bearbeiters, eine Modulzuordnung, eine Anhangszuordnung sowie eine Kategoriezuordnung.

Die Anwenderbeiträge können nach folgenden „Bedürfnissen“ unterschieden werden:

- „Fehlerbeseitigung“ („Error occurs upon Excel download.“),
- „Neue Funktion“ („New control that renders a target control to excel.”),
- „Erweiterung“ einer existierenden Funktion („Each attachment should be markable as deployment info.“),
- „Verschieben“ einer existierenden Funktion („Move the WeeklyHedgingModule“),
- „Änderung“ einer existierenden Funktion („Rename Net Debt Monitor to Financing Monitor“),
- „Entfernen“ einer existierenden Funktion („Remove all GTM references from Monthly Cash Planning“) und
- „Frage“ zu einer existierenden Funktion („[…] gehen davon aus, daß […] evtl. die Deals im Portal bestätigt hat ohne eine Confirmation erhalten zu haben. Kann das sein?“).

Im Rahmen der Analyse wurde die an der CoFiPot-Entwicklung zentral beteiligte Person befragt. Als Vorgehensweise für das Gespräch wurde das narrative Interview gewählt. Der Entwickler berichtete aus dem Gedächtnis heraus von seinen Erfahrungen und Eindrücken. Anschließend wurde ihm die Beschreibung der Anforderungen vorgelegt und er wurde um seine Meinung gebeten. Das Gespräch wurde aufgezeichnet und die Ergebnisse der Auswertungen von ihm auf Korrektheit überprüft.

Das Ergebnis des Interviews lässt sich folgendermaßen zusammenfassen: Bei der Entwicklung nahm er gleichzeitig die Rolle eines Entwicklers und die eines Entscheiders wahr. Die Befragung ergab, dass die Herstellung eines unmittelbaren Bezuges zu einem Objekt bzw. einer Funktion im Software-Produkt mit Hinweis auf den Nutzungskontextes des Vorschlages eine große Erleichterung für den Entscheider darstellte. Außerdem waren konkrete Darstellungen im Issue förderlich. Daher sollte ein Anwenderfeedback ein klar formuliertes Bedürfnis aus den Kategorien „Fehler“, „Neue Funktion“, „Erweiterung“, „Verschieben“, „Verbessern“, „Entfernen“ und „Frage“ enthalten. Auch das Vorhandensein von Bildschirmfotos, Fehlercodes und die Beschreibung von Systemparametern wurden als hilfreich angesehen. Aus Sicht dieser Person besitzt die Beschreibung der Systemparameter die größte Bedeutung, da sich daraus der Nutzungskontext des Anwenders am besten nachvollziehen lässt. Fehlercodes und Bildschirmfotos können ebenfalls nützlich sein, wobei Stichworte in der Vorschlags-

beschreibung meist ausreichen um sich im Programm zu orientieren. Für Personen, die mit dem Programm nicht vertraut sind, besitzen Bildschirmfotos und Fehlercodes eventuell eine größere Bedeutung. Hieraus lassen sich die Anforderungen A14, A15 und A16 ableiten:

A14: „Herstellung eines Bezuges" – Der Anwender kann einen unmittelbaren Bezug zu einer Funktion herstellen.

A15: „Kategorisierung der Anwenderbeiträge" - Der Anwender kann seine Anwenderbeiträge einer der folgenden Kategorien zuordnen: „Fehler in einer Funktion", „Neue Funktion", „Erweiterung einer Funktion", „Verschieben einer Funktion", „Verbessern einer Funktion", „Entfernen einer Funktion", „Frage zu einer Funktion".

A16: „Bildschirmfoto, Fehlercode und Systemparameter als Nutzungskontext" - Als Nutzungskontext der Software-Nutzung sind Bildschirmfotos, Fehlercodes und Systemparameter hilfreich und sollten bei der Erstellung von Anwenderbeiträgen beigefügt werden.

Ferner bestätigte die befragte Person die bereits identifizierten Anforderungen A7 „Integration der Entwicklungsumgebung", A11 „Erstellung von Anwenderbeiträgen im Nutzungskontext", A12 „Übersichtliche Darstellung und Anpassung der Anwenderbeiträge" und A13 „Zeitnahe Entscheidungsfindung". Die Integration der Anwenderbeiträge in die Entwicklungsumgebung ist bei Bayer bereits Realität, so dass eingehende Anwenderbeiträge direkt als Änderungsvorschlag im Issue-Tracking-System bearbeitet werden. Dies erleichtert die Verwaltung der Anwenderbeiträge und ermöglicht darüber hinaus eine zeitnahe Entscheidungsfindung, so dass zeitnah auf die Wünsche der Anwender reagiert werden kann. Aus Zeitgründen ist die Integration des Issue-Tracker in die Microsoft Visual Studio Entwicklungsumgebung noch nicht realisiert. Außerdem kann es passieren, dass das Issue-Tracking-System übergangen wird, wenn kleine Verbesserungsvorschläge im Rahmen eines Telefonates kommuniziert und auch gleich umgesetzt werden. Die Erstellung der Anwenderbeiträge erfolgt ausschließlich im Nutzungskontext, da neue Versionen der Software als Aktualisierung aufgespielt und Anwenderbeiträge direkt für den realen Betrieb abgegeben werden können. Für zusätzliche Benutzertests stehen keine finanziellen und personellen Ressourcen zur Verfügung, so dass diese Art der Anwenderbeteiligung zwar einerseits eine Art Notlösung darstellt, andererseits jedoch eine schnelle Anwender-Entwickler-Kommunikation bedeutet und als gewinnbringend wahrgenommen wird. Da die Formulierung der Anwenderbeiträge nicht immer verständlich gewählt ist, ist es wichtig, dass die Anwenderbeiträge nachträglich angepasst werden können.

6.3.2.2 Analyse der Inhalte einer Email-Kommunikation zwischen Entwicklern und Anwendern zum Entwicklungsprojekt TopBraid-Composer (TBC)

Der TopBraid-Composer (TBC) ist eine Software und Teil der TopBraid-Suite von TopQuadrant[10], ein Software-Unternehmen, das 2001 in USA gegründet wurde. TopQuadrant bietet mit seiner TopBraid-Suite eine Entwicklungsumgebung zur Implementierung von Lösungen, die auf Semantic-Web-Technologien basieren. Das Unternehmen agiert von mehreren Büros in den USA aus und hat seine Kunden in Europa und USA. Abbildung 6.8 stellt ein Bildschirmfoto von TBC in der Version 2.0 dar.

[10] TopQuadrant, http://www.topquadrant.com/company, abgerufen am 16.03.2009

Der TopBraid-Composer ist als Eclipse-Plug-In implementiert und stellt eine Entwicklungsumgebung für die Anwendung der anderen Komponenten der TopBraid-Suite zur Verfügung. Im Composer können Ontologie-Modelle zu der entwickelten Software erstellt, die Anbindung mehrerer Datenquellen konfiguriert sowie die Darstellung und Verarbeitung der Daten (z.B. bei dynamische Masken oder Berichte) angepasst werden.

Kurz vor der Vermarktung im Jahr 2006 stellte sich ein Mitarbeiter des FZI Forschungszentrum Informatik als Anwender bzw. Beta-Tester des TBC zur Verfügung. Die Kooperation kam aufgrund eines freundschaftlichen Verhältnisses zwischen dem Entwickler der Software und dem Anwender zustande. Der Anwender hatte zu diesem Zeitpunkt bereits ein Studium der Informatik absolviert und beschäftigt sich in seiner Promotion mit Semantic Web Technologien. Die Motivation des Anwenders zum Testen vom TBC war rein privater Natur, da er sich beruflich in einem ähnlichen Thema bewegte und selbst ebenfalls als Entwickler in Projekten arbeitete - allerdings nicht als Entwickler von TBC.

Die Kommunikation zwischen dem Anwender und den Entwicklern verlief ausschließlich per Email. Als Sprache wurde hauptsächlich Englisch verwendet, in manchen Fällen formulierte der Anwender seine Emails jedoch hin und wieder in deutscher Sprache. Der Anwender begründete dies damit, dass diese Mails an einen speziellen in Deutschland aufgewachsenen und in den USA lebenden Entwickler gerichtet waren. In unregelmäßigen Abständen rief der Anwender die aktuellste Version von TBC ab, installierte und testete diese und schrieb seine Kommentare per Email an die Entwickler. Von April bis Oktober 2006 sendete der Anwender über 70 Mails bezüglich TBC an die Entwickler des Programms. Die TBC-Software durchlief dabei mehrere Versionen von Version 0.81 bis Version 1.1.3.

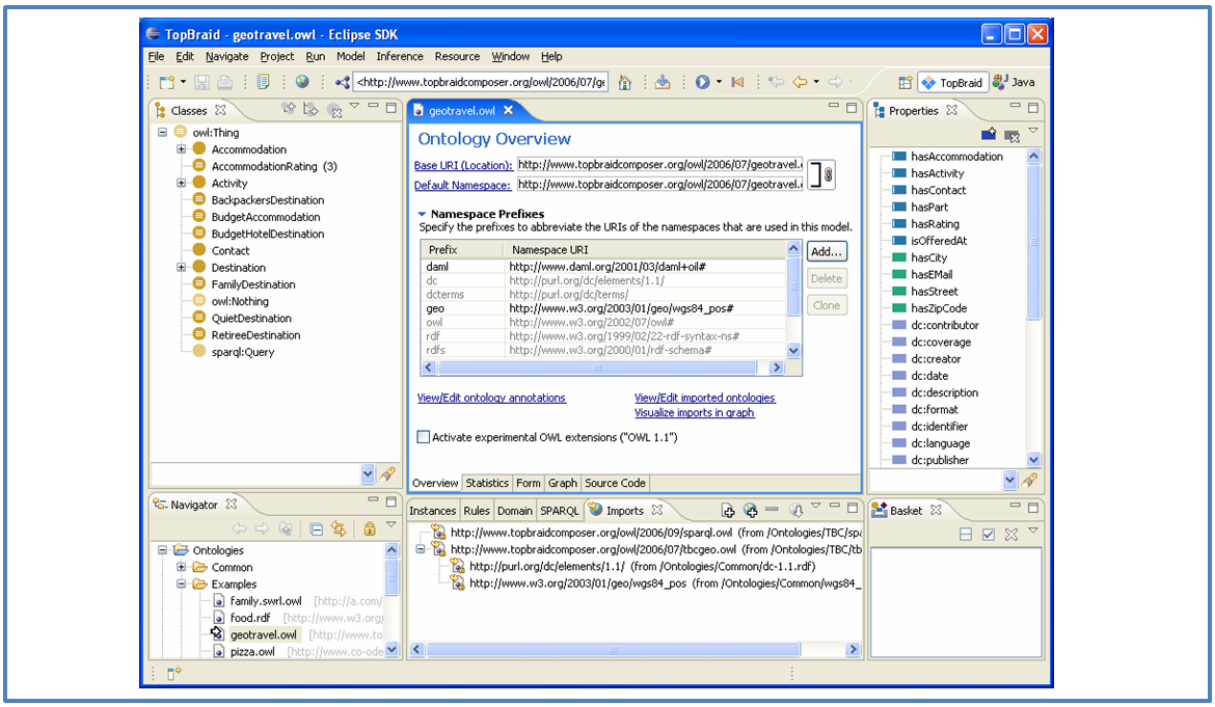

Abbildung 6.8: TopBraid-Composer (TBC) in der Version 2.0[11]

[11] Diese Abbildung stammt aus der Anleitung „TBC Getting started Guide Version 2.0“ vom 21.07.2007, die auf http://www.topquadrant.com zur Verfügung gestellt wurde, abgerufen am 16.03.2009.

Die Durchsicht der Emails lässt deutlich erkennen, dass der Anwender die Lage des Entwicklers sehr gut einschätzen konnte, da er sich selbst häufig in einer ähnlichen Situation befindet. Die Emails sind sehr ausführlich gehalten und werden im Betreff konsequent kategorisiert. Jede Nachricht weist dabei eine klare Struktur auf. Zunächst werden im Betreff in einem Satz die Art des Feedbacks sowie eine kurze Zusammenfassung der Nachricht beschrieben. Der Nachrichtentext erläutert dann zu Beginn die Problemstellung, rekonstruiert die Herangehensweise und führt schließlich Gründe für die Notwendigkeit des Vorschlages auf. Anschließend werden mögliche Lösungsmöglichkeiten aufgelistet und deren Auswirkungen aus Sicht des Anwenders abgeschätzt. In vielen Fällen wurden den Emails Bildschirmfotos und Beispieldatensätze beigefügt. Die Bildschirmfotos zeigten dabei einen Ausschnitt der Anwendung. Die Elemente, auf die der Anwender sich bezog, wurden in einem Grafikprogramm freihändig mit einem roten Stift markiert und in wenigen Fällen durch einen Text ergänzt. Zudem wurden mehrere Emails mit allgemeinem Feedback zur Entwicklung gesendet, die eine Zusammenfassung des allgemeinen Eindruckes des Software-Produkts wiedergaben und eher motivierender Natur waren.

Die Antworten auf die Emails des Anwenders erfolgten zeitnah ebenfalls per Email und enthielten entweder eine Rückfrage oder eine Entscheidung zur weiteren Vorgehensweise. Wurde ein Vorschlag abgelehnt, wurde der Grund der Entscheidung mitgeteilt. Rückfragen dienten der Klärung der Problemstellung und der Diskussion von möglichen Lösungsalternativen.

Betreff:

[TBC 1.0.0] DEFECT: Exception on pasting into field in Form view

Nachrichtentext:

---+++ DEFECT: Exception on pasting into field in Form view

I tried to copy&paste a SPARQL query directly into an rdfs:comment field in the Form view. This resulted in a NullPointerException. See attached screenshot and "log.txt". Entering the query by hand worked fine (resulted in an context menu entry "execute query").

Notice: I did not try out if this is a general problem with pasting into form fields.

Anhang:

log.txt
tbc-form-paste-npe.png

Abbildung 6.9: Email einer Fehlermeldung vom 12.05.2006 vom Anwender an den Entwickler

```
!ENTRY org.eclipse.ui 4 4 2006-05-12 12:29:35.718
!MESSAGE "Paste" did not complete normally.  Please see the log for more information.

!ENTRY org.eclipse.ui 4 0 2006-05-12 12:29:35.718
!MESSAGE java.lang.NullPointerException
!STACK 0
java.lang.NullPointerException at
org.eclipse.ui.texteditor.AbstractTextEditor$ElementStateListener$Validator.verifyText(AbstractTextEditor.jav
a:243)
                at org.eclipse.swt.widgets.TypedListener.handleEvent(TypedListener.java:194)
                at org.eclipse.swt.widgets.EventTable.sendEvent(EventTable.java:66)
[...]
```

Abbildung 6.10: Anhang „log.txt“ zur Email vom 12.05.2006

Zur Veranschaulichung der Email-Kommunikation dient als Beispiel eine Email vom 12.05.2006 (vgl. Abbildung 6.9), worin der Anwender einen Fehler an den Entwickler meldet. Im Betreff werden Name der Anwendung, die Version, die Kategorie des Feedbacks sowie eine kurze Beschreibung genannt. Im Nachrichtentext wird das Problem beschrieben und auf den Anhang verwiesen. Im Anhang befinden sich die Fehlermeldung als Textdatei (vgl. Abbildung 6.10) und ein Bildschirmfoto mit annotierten Kommentaren (vgl. Abbildung 6.11).

Die Analyse der Email-Kommunikation und das anschließende Interview mit dem Anwender bestätigen einige der bisher identifizierten Anforderungen. Die Kategorien in Anforderung A17 „Kategorisierung der Anwenderbeiträge“ sowie Anforderung A9 „Änderbarkeit der Anwenderbeiträge“ treffen zu. Der Betreff der Emails enthält stets einen Hinweis auf die Art bzw. Kategorie des Feedbacks. Als Kategorien werden die Bezeichnungen „CHANGE“, „DEFECT“, „ENHANCE“, „FEATURE“ und „CORRECTION“ verwendet. Außerdem werden einige Emails mit „IDEA“ im Betreff eingeleitet, die analog zu „FEATURE“ verwendet wurden. Die Emails, die als „CHANGE“ markiert wurden, beschreiben Anwenderbeiträge zu Änderungen des Programmverhaltens, D.h. eine bereits existierende Funktion soll geändert werden. „DEFECT“ bezeichnet Fehlermeldungen, während „ENHANCE“ Erweiterungen zu einer vorhandenen Funktion nennt. Mit „FEATURE“ bzw. „IDEA“ werden neue Funktionen vorgeschlagen. „CORRECTION“ wird verwendet, wenn der Anwender einen seiner vorherigen Beiträge korrigieren möchte. Dies war beispielsweise der Fall, als er einen Fehler meldete und ein paar Tage später feststellen musste, dass er diesen nicht mehr reproduzieren konnte. Da er seine Mail nicht rückgängig machen konnte, sendete er eine neue „CORRECTION“-Mail mit der Beschreibung seines Irrtums. In einigen Fällen wurden außerdem Fragen zu TBC gestellt, wobei im Betreff keine Kategorie vergeben, sondern lediglich einleitend mit „Frage“ in deutscher Sprache begonnen wurde. In Bezug auf die Anforderung A16 „Bildschirmfoto, Fehlercode und Systemparameter als Nutzungskontext“ ist festzustellen, dass die verwendeten Kategorien bis auf „CORRECTION“ in A16 wiederzufinden sind: „CHANGE“ = „Ändern einer Funktion“, „DEFECT“ = „Fehler in einer Funktion“, „ENHANCE“ = „Erweitern einer Funktion“, „FEATURE“ = „Neue Funktion“, „Frage“ = „Frage zu einer Funktion“. „CORRECTION“ findet sich in der Anforderung A9 wieder, die besagt, dass sich bereits abgegebenes Feedback nachträglich ändern lassen sollte. Dadurch, dass nicht nur zu Software sondern auch zur Dokumentation, der Hilfefunktion und der Installationsroutine Feedback gegeben wurde, findet wie bereits oben erwähnt auch Anforderung A2: „Plattformunabhängigkeit“ Anwendung.

Im Interview mit dem Anwender wurden die damaligen Erfahrungen besprochen und die Email-Kommunikation diskutiert. Als Vorgehensweise für das Gespräch wurde das narrative Interview gewählt. Der Anwender berichtete aus dem Gedächtnis heraus von seinen Erfahrungen und Eindrücken. Anschließend wurde ihm die Beschreibung der Anforderungen vorgelegt und er wurde um seine Meinung gebeten. Das Gespräch wurde aufgezeichnet. Die Ergebnisse der Auswertungen wurden von ihm auf Korrektheit überprüft.

Dabei wurde festgehalten, dass die zeitnahe und unkomplizierte Kommunikation mit dem Entwickler von großer Bedeutung war. Eine baldige Auseinandersetzung des Entwicklers mit den abgegebenen Anwenderbeiträgen sowie daraus resultierendem Verständnis und Umsetzung trug sehr zur Motivation des Anwenders bei. Die schnelle Reaktion war auch darauf zurückzuführen, dass der Anwender sich sehr viel Zeit nahm um das System zu verstehen und die Formulierung des Feedbacks bewusst

verständlich und präzise hielt. Der Entwickler konnte so vom Feedback des Anwenders profitierten. Den Anforderungen A4: „Leichte bzw. schnelle Verständlichkeit der Anwenderbeiträge“, A6: „Transparenz des Bearbeitungsstatus“ und A8: „Effizienz der Erstellung der Anwenderbeiträge“ wurde zugestimmt, da eine effiziente und effektive Vorgehensweise in diesem Fall sehr wichtig war. Entscheidend war dabei, dass eine Rückmeldung über die Sinnhaftigkeit der abgegebenen Anwenderbeiträge erfolgte, die Abgabe von Anwenderbeiträgen einfach und schnell per Mail und angehängte Dateien möglich war und das Bemühen um eine gute Verständlichkeit der Anwenderbeiträge unnötige Kommunikation ersparte. Zudem wurde es als notwendig erachtet, Anforderung A14 „Herstellung eines Bezuges“ zu erfüllen um auf diese Weise in den Anwenderbeiträgen einen klaren Bezug zur Software herzustellen, wodurch sich Missverständnisse vermeiden lassen.

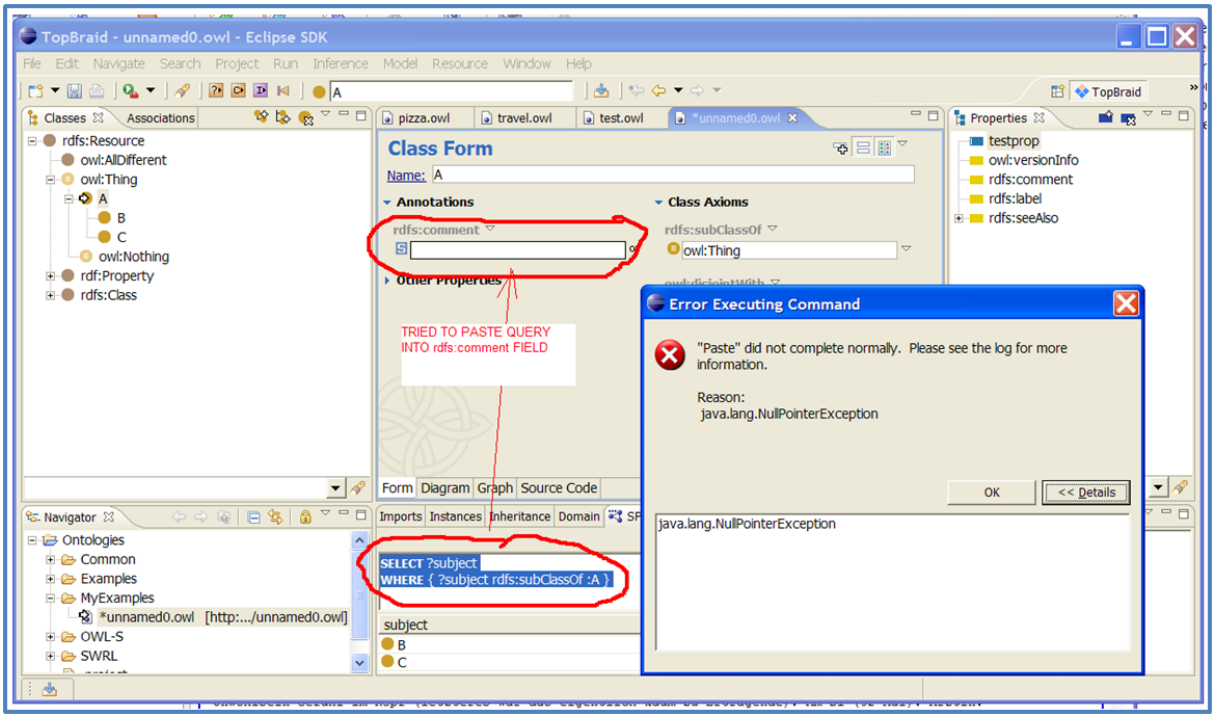

Abbildung 6.11: Anhang „tbc-form-paste-npe.png“ zur Email vom 12. Mai 2006

Es existierte in diesem Fall kein expliziter Prozess der Anwenderbeteiligung. Dem Anwender war es freigestellt, seine Anwenderbeiträge zu jedem beliebigen Zeitpunkt abzugeben, was für diesen von großer Bedeutung war, da er sich mit vielen Programmteilen oft erst später auseinandersetzen konnte und sich zunächst mit der Software vertraut machen musste. Die Möglichkeiten komplexer Software lassen sich oft erst nach ein paar Wochen Nutzung ausschöpfen und erst zu diesem Zeitpunkt werden weitere Verbesserungspotentiale erkannt. Dies bestätigt somit Anforderung A11: „Erstellung von Anwenderbeiträgen im Nutzungskontext“.

Der Nutzungskontext des Nutzers konnte leicht über Bildschirmfotos, Fehlercodes und Systemparameter nachvollzogen werden. Daher stimmte er Anforderung A16: „Bildschirmfoto, Fehlercode und Systemparameter als Nutzungskontext“ zu. Allerdings ist es notwendig, den Hergang bei einer Fehlermeldung nachvollziehen zu können, so dass auch Anforderung A5 „Erfassung des Nutzungskontextes des Anwenders“ von großer Bedeutung ist. Als wichtig wurde auch die Wahrung der Privatsphäre eingeschätzt. Daher wird auch Anforderung A10: „Vertraulichkeit“ als zutreffend bezeichnet, die

besagt, dass der Anwender bestimmen sollte, welche Informationen in den Anwenderbeiträgen enthalten sein dürfen.

Darüber hinaus konnte der Email-Kommunikation entnommen werden, dass in fast allen Fällen ein einziger Anwenderbeitrag pro Email formuliert wurde. Der Anwender bezeichnete diese Vorgehensweise als sehr hilfreich, da Diskussionen so leichter geführt und der Bearbeitungsstatus in Bezug auf die Emails ausgetauscht werden konnten. Im Falle einer Email-Kommunikation sollte pro Email immer nur ein Anwenderbeitrag versendet werden. Verallgemeinernd ist zu fordern, dass mehrere Anwenderbeiträge voneinander getrennt geführt und besprochen werden sollten. Diese Forderung ist in Anforderung A6 „Transparenz des Bearbeitungsstatus“ wiederzufinden.

6.3.2.3 Analyse der Inhalte von Anforderungsdokumenten, die im Rahmen von Usability Workshops entstanden sind (TRUMPF)[12]

Von den Unternehmen TRUMPF[13] wurden Anforderungsdokumente zur Analyse bereitgestellt, die in früheren Usability Workshops erzeugt wurden. Diese Dokumente enthalten Tabellen und Notizen, die das Ergebnis der Usability Workshops dokumentieren und als Basis für die weitere Entwicklung der TRUMPF TruTops Software-Produkte dienten. Abbildung 6.12 zeigt exemplarisch an einem Ausschnitt der Tabellendatei, in welcher Form die Anwenderbeiträge nach einem Usability Workshop bei TRUMPF vorlagen.

Die Usability Workshops wurden Ende 2006 von November bis Dezember durchgeführt. Es fanden zwei Workshops statt, bei denen jeweils ein Software-Produkt aus der TRUMPF-Software-Palette getestet wurde. Zuvor waren neue Versionen der gesamten Software-Palette fertig gestellt worden, die in den Workshops zur Diskussion gestellt werden sollten. Außerdem sollte im Rahmen des Workshops die Zeit zum Einlernen der Mitarbeiter geprüft werden, um dadurch den erforderlichen Schulungsaufwand abschätzen zu können. Der Schulungsaufwand sollten keinen zu großen Zeitaufwand für die Kunden des Unternehmens bedeuten, da es ansonsten zur Ablehnung des Produktes kommen kann.

Der erste Workshop dauerte fünf, der zweite vier Tage, wobei pro Workshop unterschiedliche Testgruppen teilnahmen. Jede Testgruppe bestand aus zehn Testpersonen. Allerdings konnten nicht alle Testpersonen an allen Tagen teilnehmen, so dass ein Workshoptag im Regelfall von sechs bis zehn Personen besucht wurde. Die Testpersonen waren Mitarbeiter von TRUMPF aus den Abteilungen „Technische Dokumentation“, „Vorführzentrum“, „Schulung“, „International“ und „Support“. Alle Testpersonen waren mit der vorherigen Version der Software vertraut. Der Verlauf des Workshops erfolgte dabei stets nach demselben Schema: Nach Begrüßung und Einführung wurden die Testpersonen am Vormittag gebeten, Testaufgaben mit Hilfe der neuen Software zu lösen. Am Nachmittag konnten die Testpersonen die Software dann dem eigenen Interesse entsprechend ausprobieren. Abschließend fand eine Feedbackrunde statt. Die Testpersonen wurden gebeten, ihr Feedback auf

---

[12] Die TRUMPF Gruppe ist weltweit führend in Maschinenwerkzeuge und Medizintechnik und hat ihren Hauptsitz in Ditzingen (Deutschland). Mit einem Umsatz von 1.65 Milliarden € beschäftigen sie global 6.500 Mitarbeiter (Stand 31.12.2007). Diese Daten wurden dem Jahresbericht 2007 der TRUMPF Gruppe entnommen.

[13] Diese Analyse erfolgte im Vorfeld der Fallstudie TRUMPF (vgl. Kapitel 9.4, Seite 177ff).

Notizzettel aufzuschreiben und vor der Feedbackrunde einen Fragebogen auszufüllen. In der Feedbackrunde konnte jede Testperson im Rahmen einer offenen Diskussion die eigene Meinung einbringen und eine Bewertung der Software abgeben. Die Notizen und Fragebögen wurden „in den Computer abgetippt“ und in eine elektronische Tabellendatei überführt. Dabei wurden identische bzw. ähnliche Vorschlägen gruppiert und die Häufigkeit ihrer Nennungen eingetragen.

Insgesamt liegen über 667 Anwenderbeiträge zur Analyse vor. Nahezu alle Anwenderbeiträge beziehen sich konkret auf eine Funktion (z.B. „Datei -> Speichern Auswahl => Keine Eigenschaften-Maske.“). Nur wenige Anwenderbeiträge waren allgemeiner Art (z.B. „Verbesserung: Das System sollte DICH führen, nicht suchen müssen.“). Ähnlich wie beim Issue-Tracker bei Bayer waren die Anwenderbeiträge oft sehr kurz gehalten und bestanden zumeist aus einem einzigen kurzen Satz. In wenigen Fällen wurde eine längere Beschreibung abgegeben.

Direkt nach den Workshops wurde mit der Bearbeitung der Anwenderbeiträge begonnen. Dabei führten die Entscheider eine Bewertung der Anwenderbeiträge durch und markierten diejenigen, die umgesetzt werden sollten. Häufig wurden die Anwenderbeiträge durch ergänzende Kommentare erweitert, mit denen die jeweilige Entscheidung begründet wurde. Dabei kommentierten die Entwickler auch unverständliche bzw. nicht reproduzierbare Anwenderbeiträge. Anwenderbeiträge, die für die Umsetzung freigegeben wurden, wurden mit einer „Erledigt“-Markierung versehen. Eine anschließende Kommunikation mit den Anwendern erfolgte nicht.

Meldung	Anzahl	Erl.	Bewertung
Es ist nicht eindeutig ob man gerade im Kopierer oder Verschieben Modus ist.	10	x	Es reicht eventuell aus, wenn verschobenes Originalteil verschwindet und zu kopierendes Teil nicht
Verbesserung: Taste für Markierung Aufheben	1	x	oder mit ESC - erstes ESC bricht Automat ab zweites entfernt Markierung.
Funktion "Markierte Teile um die vorherige Strecke bewegen! auch in Verschieben (ist nur beim Kopieren)	3	x	
Bereich automatisch belegen nur möglich, falls Teil an Maus.		x	nur enabled, wenn Teil an Cursor.
In der Materialdatenbank nur St vorhanden. Rest fehlt komplett.	1		?
Unter MTL gibt es GTS-Knöpfe.	1	x	
Text von Icon falsch. GTS 2 Punkte muss heißen Abstand 2 Punkte; GTS Drehen muss heißen Abstand Drehen. GTS Drehen hat mit GTS nichts zu tun.	1	x	
Kopieren ok, aber auch für Original verschieben.	1	x	Ändern 1 fehlt ?

Abbildung 6.12: Ausschnitt[14] aus der Dokumentation eines Usability Workshops bei TRUMPF

[14] Zwei Rechtschreibfehler wurden aufgrund der Lesbarkeit korrigiert

Aus den Kommentaren zu den Anwenderbeiträgen kann geschlossen werden, dass zu insgesamt 92 der 667 Vorschläge eine Rückfrage bei dem Anwender bzw. der Testperson hilfreich gewesen wäre, die den Anwenderbeitrag abgegeben hatte. 63 dieser Vorschläge waren „nicht selbsterklärend“ bzw. konnten nicht reproduziert werden. Die anderen 29 Vorschläge waren unklar formuliert und den Entscheidern war die genaue Absicht des Anwenders nicht klar. (z.B. „Ist auch hier die Bemaßungsfarbe gemeint?“).

Die in Anforderung A15 „Kategorisierung der Anwenderbeiträge“ aufgestellten Kategorien für Anwenderbeiträge lassen sich auch auf die TRUMPF Anwenderbeiträge übertragen. Zudem finden sich in den Beiträgen auch lobende und positive Rückmeldungen, so dass die Kategorie „Lob zu einer Funktion“ in Anforderung A15 ergänzt wird:

A15: „Kategorisierung der Anwenderbeiträge“ - Der Anwender kann seine Anwenderbeiträge zu einem der folgenden Kategorien zuordnen: „Fehler in einer Funktion“, „Neue Funktion“, „Erweiterung einer Funktion“, „Verschieben einer Funktion“, „Ändern einer Funktion“, „Entfernen einer Funktion“ , „Frage zu einer Funktion“ , „Lob zu einer Funktion“.

Im Gespräch mit den Organisatoren der Workshops, Mitarbeitern bei TRUMPF in der Abteilung „Software-Entwicklung“, die die Rolle der Moderatoren übernommen hatten, konnten zahlreiche Hinweise für Anforderungen an das Annotationssystem von OpenProposal gewonnen und eine Vielzahl der bisher identifizierten Anforderungen bestätigt sowie zwei zusätzliche Anforderungen aufgenommen werden. Als Vorgehensweise für das Gespräch wurde das narrative Interview gewählt. Die Moderatorin berichtete aus ihrem Gedächtnis heraus von ihren Erfahrungen und Eindrücken. Anschließend wurde ihr die Beschreibung der Anforderungen vorgelegt und sie um ihre Meinung gebeten. Das Gespräch wurde aufgezeichnet. Die Ergebnisse der Auswertungen wurden von ihr auf Korrektheit überprüft.

Der Anforderung A1 „Gebrauchstauglichkeit“ ist im Zusammenhang mit Anforderung A8 „Effizienz der Erstellung von Anwenderbeiträgen“ zuzustimmen, da bei der Zusammenarbeit mit den Anwendern beobachtet werden konnte, dass die Abgabe von Feedback sehr einfach und schnell gehalten werden sollte. Ansonsten ist damit zu rechnen, dass die Anwender seltener Anwenderbeiträge erstellen. Auch die Anforderung A4 „Leichte bzw. schnelle Verständlichkeit der Anwenderbeiträge“ trifft zu, da eine Kommunikation mit den Anwendern häufig lediglich im Rahmen der Usability Workshops möglich ist. Spätere Rückfragen sind nur erschwert möglich und aufgrund des zeitlichen Aufwandes für die Anwender und Moderatoren nicht wünschenswert. Daher wird auch Anforderung A8 „Effizienz der Erstellung von Anwenderbeiträgen“ als zutreffend bezeichnet. Dies deckt sich zusätzlich auch mit den Anforderungen A14 „Herstellung eines Bezuges“, A15 „Kategorisierung der Anwenderbeiträge“ und A16 „Bildschirmfoto, Fehlercode und Systemparameter als Nutzungskontext“. Es gilt zudem Anforderung A12 „Übersichtliche Darstellung und Anpassung des Feedback“, da der Moderator die Möglichkeit zur Anpassung der Anwenderbeiträge wünscht, um identische bzw. ähnliche Anwenderbeiträge zu gruppieren bzw. im Text anzupassen. Zudem trägt die Ergänzung der Anwenderbeiträge durch Kommentare der Anwender zu einem besseren Verständnis auf Seiten des Entwicklers bei. Die Anforderung A7 „Integration der Entwicklungsumgebung“ ist ebenso von großer Bedeutung.

In Anlehnung an Anforderung A7 wurde im Anschluss an die Usability Workshops Ende 2006 das Issue-Tracking-System JIRA[8] von Atlassian bei TRUMPF eingeführt, so dass die Bearbeitung der Vorschläge internetbasiert und nicht mehr dokumentenbasiert über die Tabellendatei erfolgen konnte. Für TRUMPF wäre somit eine Integration der Erstellung der Anwenderbeiträge in ihr Issue-Tracking-System wünschenswert. Zudem war ihnen die Effizienz der Vor- und Nachbereitung der Anwenderbeteiligung wichtig. Die Zeit und Ressourcen in ihren Projekten sind aus ihrer Sicht sehr knapp bemessen, so dass die Anwenderbeteiligung zwar intensiv und gründlich durchgeführt aber der administrative Aufwand so gering wie möglich gehalten werden soll. Daher waren für sie beispielsweise auch Videoanalysen keine Option, da diese einen hohen Aufwand für die Nachbereitung bedeuten, ohne dass der daraus zu erzielende Nutzen dies rechtfertigt. Das Annotationssystem soll daher den Aufwand für die Vor- und Nachbereitung verkürzen bzw. gering halten. Dies würde auch dann der Anforderung A13 „Zeitnahe Entscheidungsfindung“ entsprechen, dass die Bewertung der Vorschläge zeitnah nach einem Usability Workshop erfolgen und damit auch zeitnah mit der Umsetzung der genehmigten Vorschläge begonnen werden kann.

Darüber hinaus wurde festgehalten, dass die Erfassung der Häufigkeit der Nennungen eines identischen bzw. ähnlichen Vorschlags essentiell wäre, um die Wichtigkeit des Vorschlages abschätzen zu können. Als Erweiterung ergibt sich hieraus folgende Modifikation von Anforderung A12:

A12: „Übersichtliche Darstellung und Anpassung der Anwenderbeiträge“ - Der Moderator kann die Anwenderbeiträge der Anwender verwalten und anpassen. Die Häufigkeit identischer bzw. ähnlicher Anwenderbeiträge kann mit geringem Aufwand erfasst werden.

6.3.2.4 Zusammenfassung der Anforderungen aus der Inhaltsanalyse

Aus der Inhaltsanalyse können zusammenfassend die bisher aufgestellten Anforderungen A1 - A13 mit einer Anpassung der Anforderung A12 aus Tabelle 6.1 bestätigt und neue Anforderungen A14 und A15 ergänzt werden (vgl. Tabelle 6.2).

Nr.	Anforderung
A1	„Gebrauchstauglichkeit“ - Die Gebrauchstauglichkeit nach den Richtlinien der ISO Normreihe 9241 (EN ISO 9241) und der ISO/IEC 9126 ist sicherzustellen.
A2	„Plattformunabhängigkeit“ - Die Anwenderbeiträge sollten unabhängig von der Art und Darstellungsformat von Software formuliert werden können. Das Darstellungsformat nimmt die Form einer lauffähigen Software oder eines Dokumentes zu einer Software ein.
A3	„Elektronische Datenhaltung“ - Anwenderbeiträge sollten durchgängig in elektronischer Form gespeichert werden.
A4	„Leichte bzw. schnelle Verständlichkeit der Anwenderbeiträge“ - Der Entwickler kann die Anwenderbeiträge mit einem niedrigen Aufwand nachvollziehen.
A5	„Erfassung des Nutzungskontextes des Anwenders“ - Die Anwenderbeiträge enthalten Informationen über die Situation des Anwenders während der Erstellung der Anwenderbeiträge und helfen dem Entwickler dabei den Nutzungskontext des Anwenders nachzuvollziehen.

Nr.	Anforderung
A6	„Transparenz des Bearbeitungsstatus“ - Der Entwickler kann den Bearbeitungsstatus der Anwenderbeiträge kommunizieren. Der Anwender kann sich über den Bearbeitungsstand seiner Anwenderbeiträge informieren
A7	„Integration der Entwicklungsumgebung“ - Die Anwenderbeiträge können in vorhandene Entwicklungsumgebungen integriert werden.
A8	„Effizienz der Erstellung von Anwenderbeiträgen“ - Der Anwender kann seine Anwenderbeiträge mit geringem Aufwand und ohne Einarbeitungszeit abgeben.
A9	„Änderbarkeit der Anwenderbeiträge“ - Der Anwender kann seine Anwenderbeiträge nachträglich ändern.
A10	„Vertraulichkeit“ - Der Anwender kann seine Anwenderbeiträge auf Wunsch anonymisiert abgeben. Rückschlussmöglichkeiten auf seine Person und auf private Daten werden so weit wie möglich verhindert.
A11	„Erstellung von Anwenderbeiträgen im Nutzungskontext“ - Der Anwender kann seine Anwenderbeiträge während der alltäglichen Nutzung der Software abgeben.
A12	„Übersichtliche Darstellung und Anpassung der Anwenderbeiträge“ - Der Moderator kann die Anwenderbeiträge der Anwender verwalten und anpassen. Die Häufigkeit identischer bzw. ähnlicher Anwenderbeiträge kann mit geringem Aufwand erfasst werden.
A13	„Zeitnahe Entscheidungsfindung“ - Der Entscheider erhält das Feedback zeitnah nach der Feedbackabgabe.
A14	„Herstellung eines Bezuges“ – Der Anwender kann einen unmittelbaren Bezug zu einer Funktion herstellen.
A15	„Kategorisierung der Anwenderbeiträge“ - Der Anwender kann seine Anwenderbeiträge einer der folgenden Kategorien zuordnen: „Fehler in einer Funktion“, „Neue Funktion“, „Erweiterung einer Funktion“, „Verschieben einer Funktion“, „Verbessern einer Funktion“, „Entfernen einer Funktion“, „Frage zu einer Funktion“ , „Lob zu einer Funktion“.
A16	„Bildschirmfoto, Fehlercode und Systemparameter als Nutzungskontext“ - Als Nutzungskontext der Software-Nutzung sind Bildschirmfotos, Fehlercodes und Systemparameter hilfreich und sollten bei der Erstellung der Anwenderbeiträge beigefügt werden.

Tabelle 6.2: Zusammenfassung der Anforderungen an das OpenProposal Annotationssystemen

Tabelle 6.3 stellt den Zusammenhang zwischen den Zielen von OpenProposal und den Anforderungen an das OpenProposal Annotationssystem her.

Nr.	Ziele von OpenProposal	Anforderungen an das Annotationssystem
Z1	Vereinfachung der Vorschlagserstellung	A1, A2, A3, A8, A9, A10, A11, A14, A16
Z2	Formale Strukturen zur besseren Verständlichkeit	A1, A5, A14, A15
Z3	Vereinfachung der Vor- und Nachbereitung	A1, A2, A3, A4, A6, A7, A12, A13, A15

Tabelle 6.3: Zusammenhang zwischen Anforderungen und Ziele von OpenProposal

6.3.3 Anpassung der Lösungskonzepte

Anhand der oben identifizierten Anforderung A10 und der Kategorisierung in Anforderung A15 ergibt sich die Notwendigkeit zur Anpassung des OpenProposal Lösungskonzeptes.

Die Sicherstellung der Vertraulichkeit in Anforderung A10 ist für die Akzeptanz der Anwender von essentieller Bedeutung. Um dem Anwender ein Gefühl der Sicherheit zu geben und einem späteren Missbrauch vorzubeugen, ist sicherzustellen, dass der Anwender keine persönlichen bzw. privaten Daten im Feedback hinterlässt (Anforderung A10) und die Speicherung der Informationen transparent nachvollziehen kann (Anforderung A6). Bei der Annotation von Bildschirmfotos besteht die Gefahr, dass private Daten auf dem Bildschirmfoto aufgenommen werden, die für die Allgemeinheit nicht bestimmt sind. Daher ist sicherzustellen, dass der Anwender das Bildschirmfoto auf den wesentlichen Ausschnitt reduzieren oder die privaten Bereiche unwiderruflich aus dem Bildschirmfoto entfernen kann. Hierfür sind, wie Abbildung 6.13 dargestellt, die Schaltfläche „Bildschirmfoto anpassen“ vorgesehen.

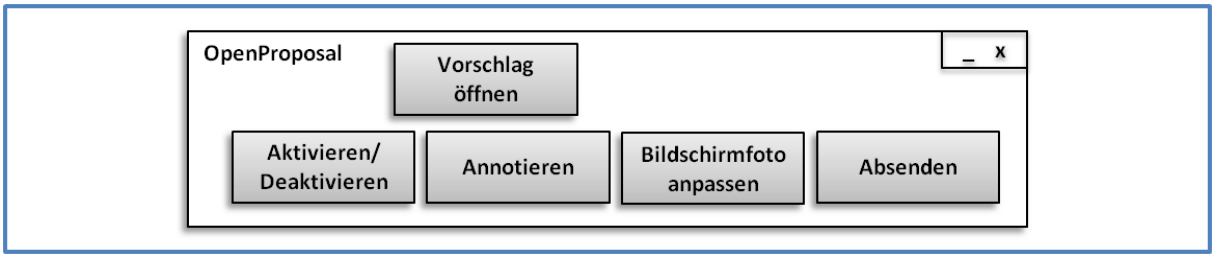

Abbildung 6.13: Erweiterung des Konzepts der Hauptbedienelemente des Annotationssystems

Zur Sicherstellung der Transparenz der Datenspeicherung kann der Anwender seine Vorschläge mit der Schaltfläche „Vorschlag öffnen“ nachträglich einsehen und sich einen Überblick über die in seinen abgesendeten Anwenderbeiträgen vorhandenen Informationen verschaffen.

Die Kategorisierung, wie sie in Anforderung A15 empfohlen wird, erfolgt in OpenProposal durch die Wahl des Annotationswerkzeuges. Das Lösungskonzept zum Annotationssystem lässt sich daran orientierend erweitern. In Abbildung 6.14 umfasst die Schaltfläche „Annotieren“ in OpenProposal acht verschiedene Kategorien, so dass der Anwender aus einer Palette von Annotationswerkzeugen wählen kann, welche Art von Vorschlag er abgeben möchte. - Außerdem ist bei der Speicherung von Systemparametern eine transparente Vorgehensweise zu wählen und das Einverständnis des Anwenders einzuholen. Über eine Konfigurationsmöglichkeit soll dem Anwender die Möglichkeit gegeben werden, die Speicherung der Systemparameter nach seinem Ermessen einzustellen. Hierfür ist die Schaltfläche „Konfigurieren“ vorgesehen.

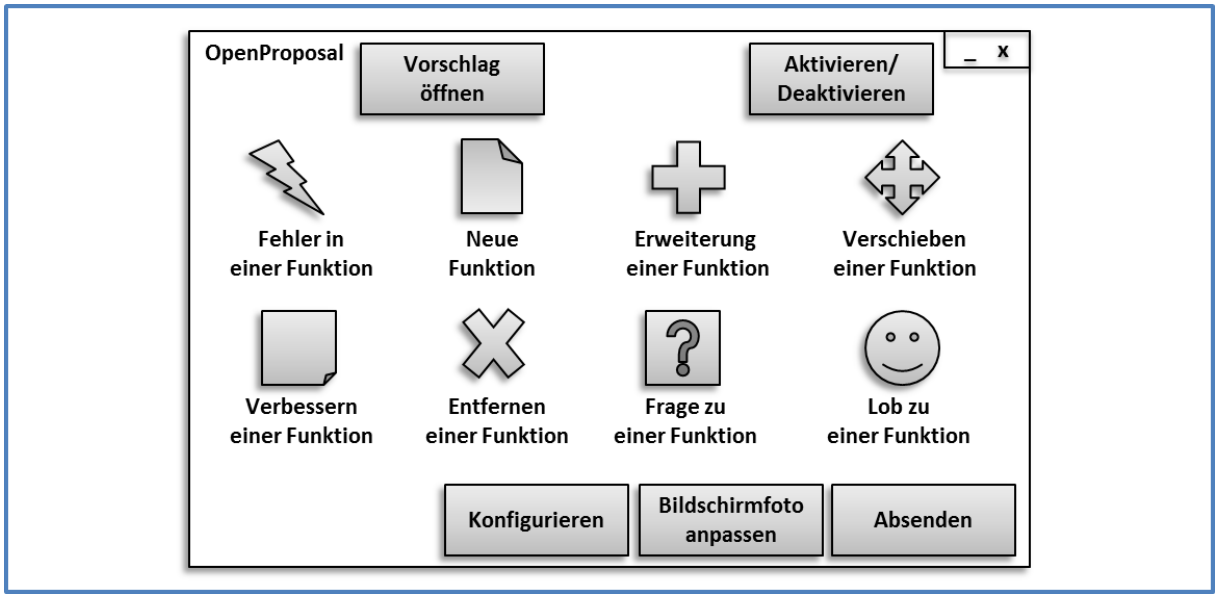

Abbildung 6.14: Konzept der Annotationswerkzeuge des Annotationssystems

Für jedes einzelne Annotationswerkzeug wird eine Anleitung zur Formulierung seines Vorschlages hinterlegt, um dem Anwender in Abhängigkeit von der Kategorie eine Vorlage für die Beschreibung seines Vorschlags anzubieten:

- Mit der Schaltfläche „Hinzufügen einer Funktion“ markiert der Anwender eine Stelle, an der die neue Funktion erscheinen soll. Hierzu sollte er einen Titel und eine kurze Funktionsbeschreibung abgeben. Idealerweise sollte der Anwender ein Beispiel angeben.

- Die Schaltfläche „Verschieben einer Funktion“ erlaubt es dem Anwender, zunächst den zu verschiebenden Bereich zu markieren und anschließend zu definieren, wohin dieser verschoben werden soll.

- Bei Auswahl der Schaltfläche „Entfernen einer Funktion“ kann der Anwender einen Bereich markieren, der entfernt werden soll. Anschließend wird er gebeten, seinen Wunsch kurz zu begründen.

- Mit der Schaltfläche „Fehler einer Funktion“ kann der Anwender den Fehler-Bereich markieren und eine Fehlerbeschreibung textlich ergänzen. Dabei wird ihm empfohlen, auch den Hergang zu beschreiben, der zum Fehler geführt hat, und seine letzten Aktionen aufzulisten, um den Fehler reproduzieren zu können. Dem Anwender wird zudem kommuniziert, dass der Fehlercode bzw. die Fehlernachricht im Idealfall im Bildschirmfoto enthalten sein sollten.

- Bei Drücken der Schaltfläche „Verbessern einer Funktion“ wird der Anwender gebeten die zu erweiternde bzw. verbessernde Funktion zu markieren und seinen Erweiterungs- bzw. Verbesserungswunsch zu artikulieren.

- Die Schaltfläche „Frage zu einer Funktion“ gestattet es dem Anwender, einen gewünschten Bereich zu markieren und hierzu eine Frage zu formulieren. Er wird dabei dazu angeleitet, seine Frage so präzise wie möglich zu formulieren und gegebenenfalls die zu Grunde liegende Motivation zu erläutern.

- Nach Auswahl der Schaltfläche „Lob zu einer Funktion“ hat der Anwender die Möglichkeit den Bereich, der ihm gut gefällt, zu markieren und lobende Worte zu wählen. Bei der Beschreibung des Lobs sollte der Anwender auch den Grund seines Lobs angeben.

- Für Fälle, bei denen der Anwender seinen Vorschlag keiner der oben genannten Kategorie zuordnen kann, kann er die Schaltfläche „Kommentar zu einer Funktion“ betätigen. Da diese Kategorie allgemeiner Natur sein kann, wird der Anwender aufgefordert, das betroffene Element der Software zu markieren und sich bei der Beschreibung seines Vorschlags so konkret wie möglich und so ausführlich wie notwendig zu äußern.

6.4 Prozessmodell

Für die Gestaltung von OpenProposal ist die flexible Integration in die Entwicklungsprozesse von Software von großer Bedeutung. Es kann nicht davon ausgegangen werden, dass Unternehmen etablierte Methoden und Prozesse neu gestalten und einen vollständig neuen Prozess installieren werden (vgl. [Gulliksen et al., 2003; Iivari und Iivari, 2006]). Die Erfahrungen der letzten Jahre mit Anwenderbeteiligung in Software-Projekten zeigen, dass die meisten Unternehmen einzelne vielversprechende Techniken und Methoden herausgreifen und in ihre bestehenden Prozessabläufe integrieren (vgl. [Iivari und Iivari, 2006; Walldius et al., 2009]). Eine radikale Umstrukturierung sollte nicht notwendig sein. Mit OpenProposal soll daher eine Methode vorgegeben werden, die sich in vorhandene Vorgehensmodelle und Unternehmensprozesse integrieren lässt.

Im Folgenden werden die Einsatzmöglichkeiten von OpenProposal in Software-Projekten erläutert und es wird ein Prozessmodell für die asynchrone Anwenderbeteiligung mit OpenProposal erarbeitet.

6.4.1 Einsatzmöglichkeiten von OpenProposal

Die asynchrone Anwenderbeteiligung mit einem Annotationssystem bedingt das Vorhandensein einer Software bzw. eines grafischen Artefaktes, welches die Software repräsentiert. In frühen Phasen der Software-Entwicklung liegt allerdings noch keine lauffähige Test-Software vor, so dass hierzu Alternativen in Betracht gezogen werden müssen. Auch wenn OpenProposal primär zum Test von lauffähiger Software bzw. Prototypen konzipiert wird, bietet das System die Möglichkeit, auch solche Repräsentationen zu annotieren, die sich noch in der Entwicklung befinden.

Wie in Abbildung 6.15 dargestellt, existieren in den einzelnen Projektphasen zu unterschiedlichen Zeitpunkten gewisse Artefakte, die mit dem Annotationssystem von OpenProposal annotiert werden können:

- Soll ein bestehendes Software-Produkt im Rahmen des Software-Projektes verbessert bzw. neu entwickelt werden, können zu Beginn der Anforderungsanalyse die im Betrieb befindlichen Software-Produkte mit Anwendern diskutiert und Verbesserungsmöglichkeiten identifiziert werden. Möglicherweise können auch vergleichbare Software-Produkte anderer Hersteller getestet werden.

- Ist eine Neu-Entwicklung geplant und existiert eine vergleichbare Software bzw. steht zum Test zur Verfügung, können Mock-Ups und elektronische Papierprototypen als Artefakte verwendet werden.

- Während der Phase des Entwurfs und der Implementierung kann der Anwender die Entwürfe zu den grafischen Benutzeroberflächen (GUI) begutachten. Sobald ein lauffähiger Prototyp existiert, können erste Anwendertests auch an lauffähiger Software durchgeführt werden.

- Sobald erste Versionen der Software vorliegen, kann die Software mit OpenProposal sowohl in der Frühentwicklung (Alpha- und Beta-Stadium), als auch in der vor-finalen Version oder im Betrieb getestet werden.

Die sequentielle Darstellung der Projektphasen in Abbildung 6.15 ist dem Umstand der Übersichtlichkeit geschuldet. Der Einsatz von OpenProposal ist nicht nur auf das Wasserfallmodell beschränkt sondern kann auch bei inkrementellen bzw. iterativen Vorgehensmodellen oder agile Methoden eingesetzt werden. Dabei sind die gleichen Artefakte wie oben beschrieben für OpenProposal geeignet.

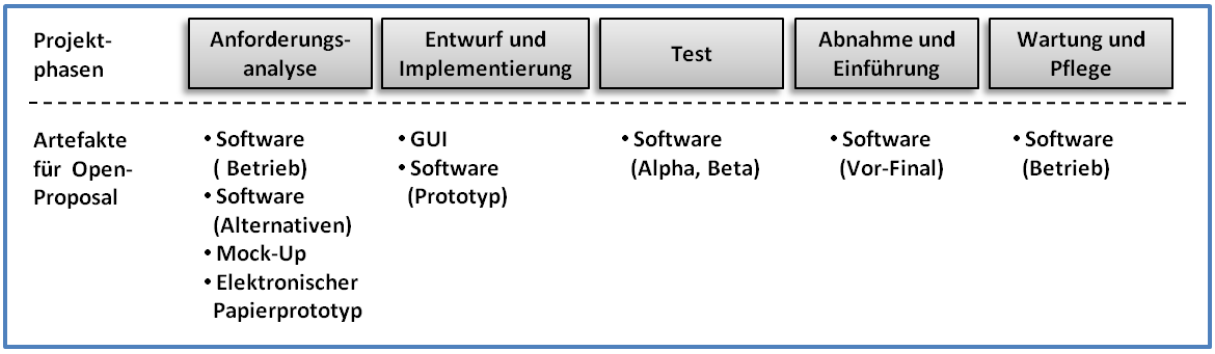

Abbildung 6.15: Einsatzmöglichkeiten von OpenProposal

6.4.2 Stufen und Schritte des OpenProposal Prozesses

Der Prozess der Anwenderbeteiligung mit OpenProposal ist in Abbildung 6.16 schematisch dargestellt. Er gliedert sich in fünf Stufen. In der ersten Stufe geben die Moderatoren den Test einer Software frei und motivieren die Anwender in Stufe 2 dazu, Anwenderbeiträge zu erstellen. Anschließend konsolidieren sie die abgegebenen Beiträge und bereiten sie für die Entscheider auf. Die Entscheider sind für die Freigabe der Beiträge verantwortlich und stimmen die weitere Entwicklung und Umsetzung der Beiträge mit den Entwicklern ab.

In Abbildung 6.17 ist der Prozess im Detail dargestellt. In Stufe 1 wird die Testumgebung durch den Moderator bereitgestellt und der Anwender zum Test eingeladen. Um die Testumgebung vorzubereiten, müssen das Annotationssystem von OpenProposal für den Test konfiguriert und die zu testenden Anwendungen für die Anwender zugänglich gemacht werden. Anschließend können die Anwender mit dem Test beginnen und Verbesserungsvorschläge erstellen. Sobald eine Verbesserungsmöglichkeit identifiziert wurde, aktiviert der Anwender das Annotationssystem, annotiert seinen Beitrag auf der Anwendung und versendet diesen (Stufe 2). Sollte er während der Annotation seines Beitrags zur Anwendung zurückkehren wollen, kann das Annotationssystem deaktiviert werden. Der Moderator sichtet anschließend in Stufe 3 die abgegebenen Anwenderbeiträge, passt diese bei Bedarf nach-

träglich an und stellt dem Entscheider die aus seiner Sicht sinnvollen Anwenderbeiträge zusammen. Der Entscheider kann die Anwenderbeiträge bei Bedarf wiederum ebenfalls nachträglich anpassen, spezifische Kommentare für die Entwickler ergänzen und die aus seiner Sicht sinnvollen Vorschläge für die Umsetzung freigeben (Stufe 4). Der Entwickler kann in Stufe 5 die Umsetzung der Anwenderbeiträge einleiten. Bei lauffähiger Software müssen die Änderungen an der Software und den Anforderungs- bzw. Entwurfsdokumenten vorgenommen werden. Bei den anderen Artefakten sind die Änderungen an den Artefakten und die zugrunde liegenden Anforderungs- bzw. Entwurfsdokumente erforderlich. Optional kann der Test an der angepassten Anwendung wiederholt werden. Auch wenn die Testumgebung dabei gleich bleibt, sollte vom Moderator hierfür eine erneute Einladung erfolgen, um die Anwender zu informieren und zum erneuten Testen zu motivieren.

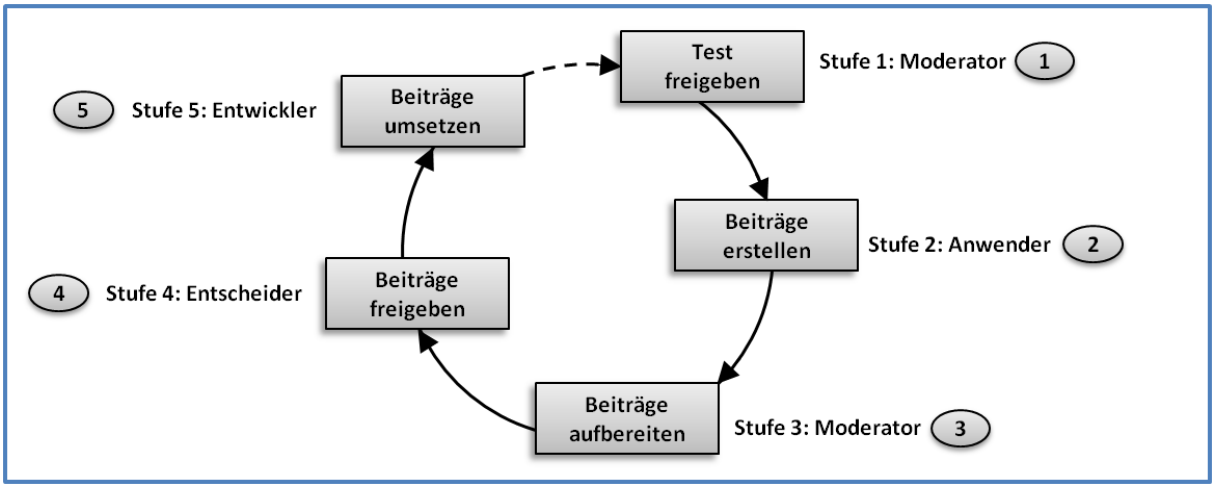

Abbildung 6.16: Stufen des OpenProposal Prozessmodells

Die Kommunikation des Bearbeitungsstatus an den Anwender wird zwar in Abbildung 6.17 aus Gründen der Lesbarkeit nicht explizit aufgeführt, ist aber ist ein essentieller Motivationsfaktor der Anwenderbeteiligung und wird im Prozess berücksichtigt. Daher wird der Bearbeitungsstatus der Anwenderbeiträge einschließlich Zulassung, Freigabe bzw. Umsetzung auf den Stufen 3, 4 und 5 beim Wechsel zur jeweils nächsten Stufe automatisch festgehalten und an den betroffenen Anwender kommuniziert. Wird ein Anwenderbeitrag mit anderen Anwenderbeiträgen zusammengefasst oder nicht weiter beachtet, wird dies dem Anwender ebenfalls gemeldet.

Die sequentielle Darstellung der Vorgehensweise soll den prinzipiellen Ablauf illustrieren. Zwischen den Stufen können sich Rückfragen und Wiederholungen ergeben, so dass der Prozess an die Gegebenheiten des Projektes angepasst werden muss. Beispielsweise können die erfassten Anwenderbeiträge unerwartete Fragen aufwerfen, was eine Rückmeldung zum Anwender und eine wiederholte Beschreibung der Vorschläge nach sich zieht.

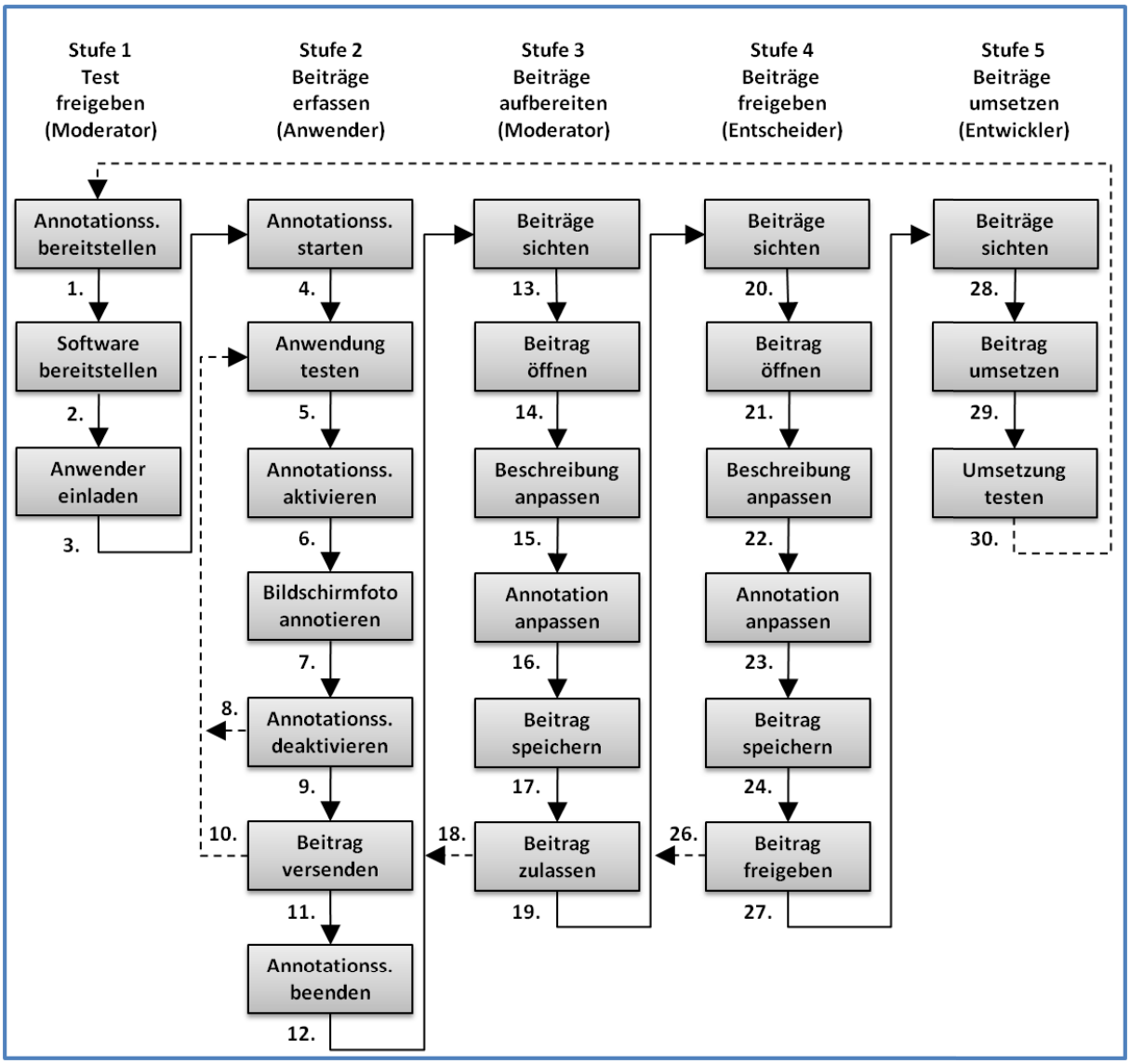

Abbildung 6.17: Schritte im OpenProposal Prozess[15]

6.5 Fazit zu Kapitel 6

Ziel von Kapitel 6 war die Konzeption der OpenProposal Methode und des OpenProposal Annotationssystems. Hierfür wurden initial Erwartungen aus der Praxis erhoben, Ziele abgeleitet und Lösungskonzepte entworfen. Anschließend wurden anhand der Analyse von Literatur und Projektdaten Anforderungen an das Annotationssystem definiert und daran anknüpfend die vorher definierten Ziele und Lösungskonzepte reflektiert und überarbeitet. Für die Beschreibung der Kommunikationswege zwischen Anwendern, Moderatoren, Entscheider und Entwickler und der Einsatzmöglichkeiten von OpenProposal in den unterschiedlichen Phasen von Software-Projekten wurde ein Prozessmodell erstellt.

[15] Die Abkürzung „Annotationss.“ steht aus Platzgründen für „Annotationssystem“.

Zentraler Bestandteil der OpenProposal Methode stellt die Idee der Annotation von Bildschirmfotos dar, die Anwendern zu einer einfachen und schnellen Erstellung von qualitativ hochwertigen Anwenderbeiträgen verhelfen soll. Dabei soll auch gemäß den Erwartungen der Praxis eine hohe Effizienz der Verwaltung der Anwenderbeiträge durch die unterschiedlichen Speichermöglichkeiten des Annotationssystems erzielt werden.

Der eigene Beitrag dieses Kapitels liegt dabei in der systematischen Anforderungsanalyse und der Konzeption der neuen OpenProposal Methode zur asynchronen Anwenderbeteiligung. Die Anforderungen an das OpenProposal Annotationssystem und die Lösungskonzepte sowie das Prozessmodell für die OpenProposal Methode stellen ein vollständiges Konzept dar, das die bisherigen Überlegungen zur Gestaltung von Methoden zur Anwenderbeteiligung berücksichtigt und einen neuartigen Ansatz bildet, mit dem eine asynchrone Anwenderbeteiligung mit dem gewünschten Erfolg ermöglicht werden soll.

Ein weiter Beitrag dieses Kapitels stellt die Erarbeitung der Kategorien von Anwenderbeiträgen dar, die in dieser Ausführlichkeit in der Literatur noch nicht existiert. Aufbauend auf die Inhaltsanalyse einer großen Menge an Anwenderbeiträgen bei einem Unternehmen und einem Open-Source-Software-Projekt wurde eine initiale Sammlung an Kategorien erstellt und im Rahmen von Befragungen erweitert und verifiziert.

Im folgenden Kapitel wird darauf aufbauend die Implementierung des OpenProposal Annotationssystems beschrieben, um die Machbarkeit der entwickelten Konzepte aufzuzeigen und die Basis für die Evaluation im Rahmen von Usability-Tests und realen Software-Projekten legen zu können.

7 Implementierung des OpenProposal Annotationssystems

Ziel dieses Kapitels ist die Beschreibung der Implementierung des OpenProposal Annotationssystems. Das Kapitel beginnt mit einem Überblick über die Systemarchitektur und die detailliierte Beschreibung der einzelnen Komponenten. Das zugrunde liegende Datenmodell erläutert die Struktur der mit OpenProposal erstellten Anwenderbeiträge sowie weitere zugehörige Daten. Anhand des Bedienkonzeptes und der Bedienoberfläche wird die grafische Benutzeroberfläche von OpenProposal vorgestellt.

Die Entwicklung von OpenProposal folgte den Prinzipien einer objektorientierten Gestaltungsstrategie („Object-Oriented Design OOD“, vgl. [Booch, 1994; Booch et al., 2007; Sommerville, 1996, S. 215]). In einem iterativen Prozess durchlief OpenProposal während seiner Entwicklung mehrere Planungs-, Implementierungs- und Testphasen. Insgesamt belief sich die Entwicklungszeit auf über zwei Jahre, in denen gemeinsam mit Benutzern und wissenschaftlichen Hilfskräften[16] Diskussionen geführt, Entscheidungen getroffen, die Umsetzungen durchgeführt und das System getestet wurden. Tests fanden in Form von Funktions-, Integrations- und Benutzertests statt. Soweit möglich, wurde das Annotationssystem auch zur Kommunikation innerhalb des Entwicklerteams verwendet, um eigene Erfahrungen damit zu sammeln.

Aufgrund der Übersichtlichkeit und Lesbarkeit werden im Folgenden die Ergebnisse der Iterationen nach ihrem Ergebnistyp dargestellt und nicht in der zeitlichen Reihenfolge. Für eine Übersicht über den Reifungsprozess von OpenProposal ist auf den Anhang zu verweisen, der die unterschiedlichen Versionen auflistet und die erfolgten Anpassungen kommentiert.

7.1 Systemarchitektur von OpenProposal

Für die Implementierung von OpenProposal wurde eine komponentenbasierte und objektorientierte Entwicklungsmethodik nach Booch et al. [2007] gewählt. Abbildung 7.1 beschreibt den Aufbau der Systemarchitektur von OpenProposal. Im Modul „OpenProposal“ befinden sich die Komponenten zur Annotation von Bildschirmfotos. Das Modul „OpenProposal Datenmanager“ steuert das Speichern der Vorschläge und integriert externe Module. Das zugrunde liegende Datenmodell zu OpenProposal wird gesondert in Kapitel 7.2 behandelt. Zu Beginn der Entwicklung wurde aus Gründen der Plattformunabhängigkeit die Programmiersprache Java[17] von Sun Microsystems als Entwicklungsumgebung eingesetzt. Allerdings stellte sich heraus, dass zum damaligen Zeitpunkt das Auslesen von Sys-

[16] Ein großer Dank gebührt an dieser Stelle Frau Monika Tavas und den Herren Jan Baumann, Florian Gand, David Meder, Carsten Rehders, und Herbert Schäfler, die als studentische Hilfskräfte maßgeblich an der Implementierung von OpenProposal beteiligt waren.

[17] SUN Java SE, http://java.sun.com/javase, abgerufen am 07.03.2009

temparametern zur Erfassung des Anwendungskontextes mit Java auf Microsoft Windows Plattformen stark eingeschränkt war. Die gewünschte Funktionalität wäre mit Java nicht erreicht worden. Daher wurde die objektorientierte Programmiersprache C#[18] bzw. Visual C# von Microsoft mit Microsoft .NET Framework[19] in der Version 2.0 als Entwicklungsumgebung gewählt. C# ähnelt Java in vielen Aspekten. Abgesehen von der Abhängigkeit von Windows Plattformen konnten keine Nachteile festgestellt werden. Als Entwicklungswerkzeug wurde Visual Studio 2005 bzw. 2008[20] verwendet. Die in Visual Studio mitgelieferten Standard-Klassen-Bibliotheken des .NET Framework (u.a. System, System.ComponentModel, System.Data, System.Drawing, System.IO, System.Network, System.Web.Services, System.Text, System.Threading, System.Windows, System.Xml) wurden bei der Realisierung von OpenProposal verwendet.

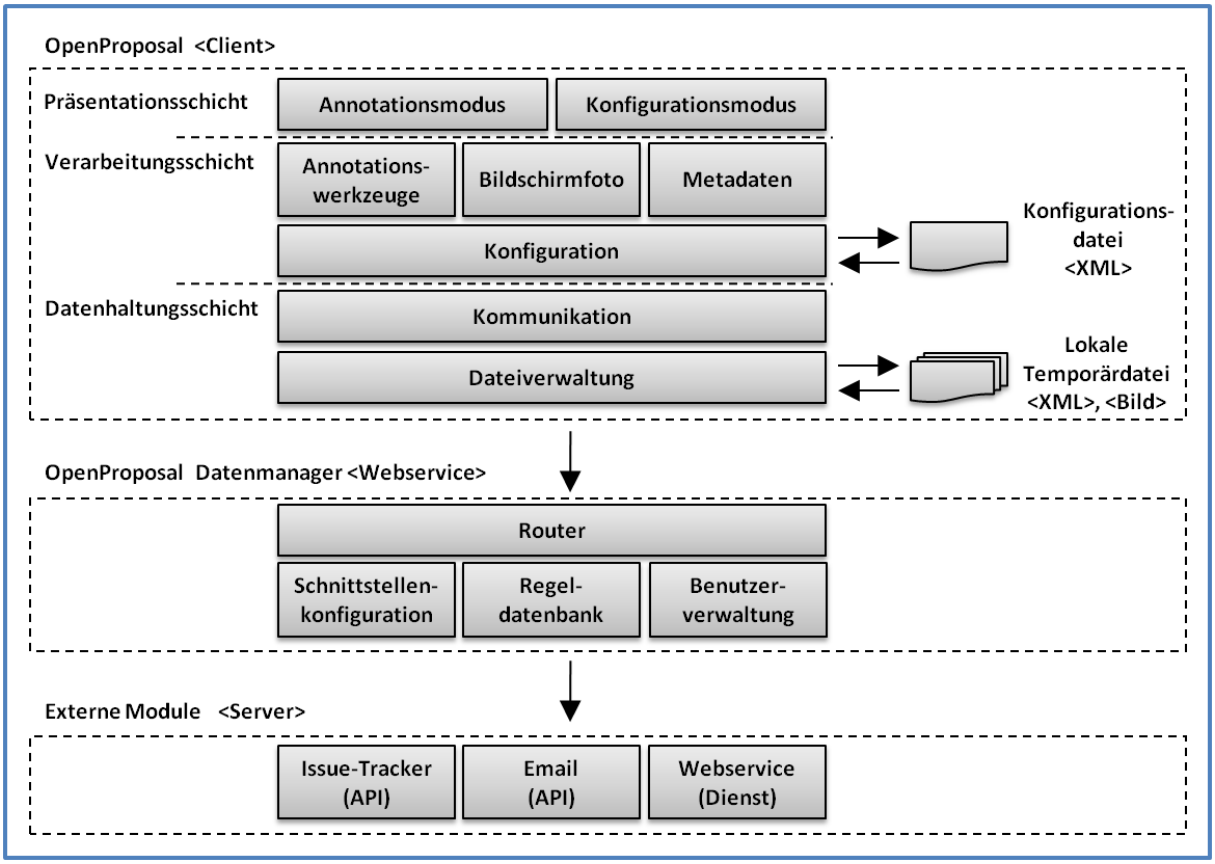

Abbildung 7.1: Systemarchitektur von OpenProposal

7.1.1 OpenProposal <Client>

Das Modul „OpenProposal“ wird dem Anwender als Client auf einer Arbeitsstation zur Verfügung gestellt. Es gliedert sich in die drei Schichten „Präsentationsschicht“, „Verarbeitungsschicht“ und „Datenhaltungsschicht“.

---

[18] Microsoft Visual C#, http://msdn.microsoft.com/de-de/vcsharp, abgerufen am 07.03.2009

[19] Microsoft .NET Framework, http://msdn.microsoft.com/de-de/netframework, abgerufen am 07.03.2009

[20] Microsoft Visual Studio, http://msdn.microsoft.com/de-de/vstudio, abgerufen am 07.03.2009

Der Anwender erhält auf der Ebene der Präsentationschicht die Möglichkeit, die Bildschirmfotos im Annotationsmodus mit den verfügbaren Annotationswerkzeugen zu annotieren. Über den Konfigurationsmodus kann der Anwender die Einstellungen von OpenProposal anpassen. Außerdem enthalten beide Komponenten die für die Gebrauchstauglichkeit notwendigen Dialoge und Assistenzfunktionen zur Interaktion mit dem Benutzer.

Auf Ebene der Verarbeitungsschicht interagieren vier Komponenten miteinander. Für die grafische Funktionalität von OpenProposal ist die Komponente „Annotationswerkzeuge“ zuständig. Unter anderem beinhaltet sie Funktionen zur Zeichnung und Positionierung der Annotationselemente auf dem Bildschirm. Die Komponente „Bildschirmfoto“ übernimmt die Verarbeitung des Bildschirmfotos und enthält Funktionen zur Erfassung, Anzeige und Speicherung eines Bildschirmfotos. Zur Verarbeitung der Metadaten eines Anwenderbeitrags, wie z.B. Titel, Beschreibung, Benutzername, Datum, Betriebssystem, Projekt etc., dient die Komponente „Metadaten“. Die Komponente „Konfiguration“ verwaltet die vom Benutzer vorgenommenen Einstellungen. Diese Daten werden in einer separaten Datei in einem XML-Format gespeichert, um die Konfiguration von OpenProposal auch nach der Kompilierung des Quellcodes vornehmen zu können. Auf die Ablage der Daten in der Registry des Betriebssystems wurde bewusst verzichtet, um den dafür notwendigen Installationsprozess zu vermeiden.

Auf der Datenhaltungsschicht stellen die Komponenten „Dateiverwaltung“ und „Kommunikation“ die Funktionalitäten zur Speicherung und Versenden des gesamten Anwenderbeitrags zur Verfügung. Während der Benutzung von OpenProposal werden die eingegebenen Daten, das Bildschirmfoto und die Metadaten in einer XML-Datei temporär zwischengespeichert. Zusätzlich wird das Bildschirmfoto mit den Annotationen als binäre Bild-Datei gespeichert.

7.1.2 OpenProposal Datenmanager

Zum Absenden der Anwenderbeiträge wird der Vorschlag lokal auf einem Laufwerk der Arbeitsstation gespeichert und im Falle der Anbindung an einen Issue-Tracker, Email oder Webservice an das Modul „OpenProposal Datenmanager“ in Form eines Webservice-Aufrufs an diese übergeben. Dieses Modul ist für die Anbindung von OpenProposal an externe Module zuständig und soll die mit OpenProposal formulierten Vorschläge in die Arbeitsumgebung der Moderatoren, Entscheider und Entwickler übertragen. Daher wurde eine allgemeine Anbindung an Issue-Tracker, Email-Postfächer und Webservice implementiert. Der Router bildet die Hauptkomponente des Datenmanagers, der die Anwenderbeiträge aufbauend auf vordefinierte Schnittstellen, Vermittlungsregeln und Zugangsdaten an externe Module weiterleitet. In der Komponente „Schnittstellenkonfiguration“ werden die Schnittstellen zu den externen Modulen verwaltet. Die Komponente „Regeldatenbank“ beinhaltet Regeldefinitionen, anhand derer die Anpassung der Speicherung der Anwenderbeiträge konfiguriert werden kann. Mit Hilfe der Benutzerverwaltung werden die im Annotationswerkzeug konfigurierten Benutzerdaten mit denen der externen Module abgeglichen.

7.1.3 Externe Module

Als externe Module werden Issue-Tracker, Email-Systeme und Webservices eingebunden. Die Anbindung erfolgt über die von den externen Modulen bereitgestellte Dokumentation bzw. Application Programming Instruction (API).

Die Integrationsmöglichkeit eines Issue-Trackers ist von der angebotenen API abhängig. Da keine Standards für die API von Issue-Trackern existieren, ist für jede Issue-Tracker-Variante eine spezifische Anbindung im OpenProposal Datenmanager zu implementieren. Im Rahmen der OpenProposal Studien wurden die Issue-Tracker Codebeamer von Intland[21], JIRA von Atlassian[Fehler! Textmarke nicht definiert.], Polarion ALM von Polarion[22] und der Open-Source-Software Issue-Tracker GForge[23] angebunden. Abbildung 7.2 illustriert den generellen Ablauf der Übertragung eines mit OpenProposal erstellten Anwenderbeitrags an einen Issue-Tracker mithilfe eines Sequenzdiagramms. Hierbei authentifiziert sich der OpenProposal Datenmanager über die API des Issue-Tracker mit dem Benutzernamen des Anwenders, erstellt einen Issue und hängt den Anwenderbeitrag als XML-Datei und Bilddatei in Form eines Anhangs an.

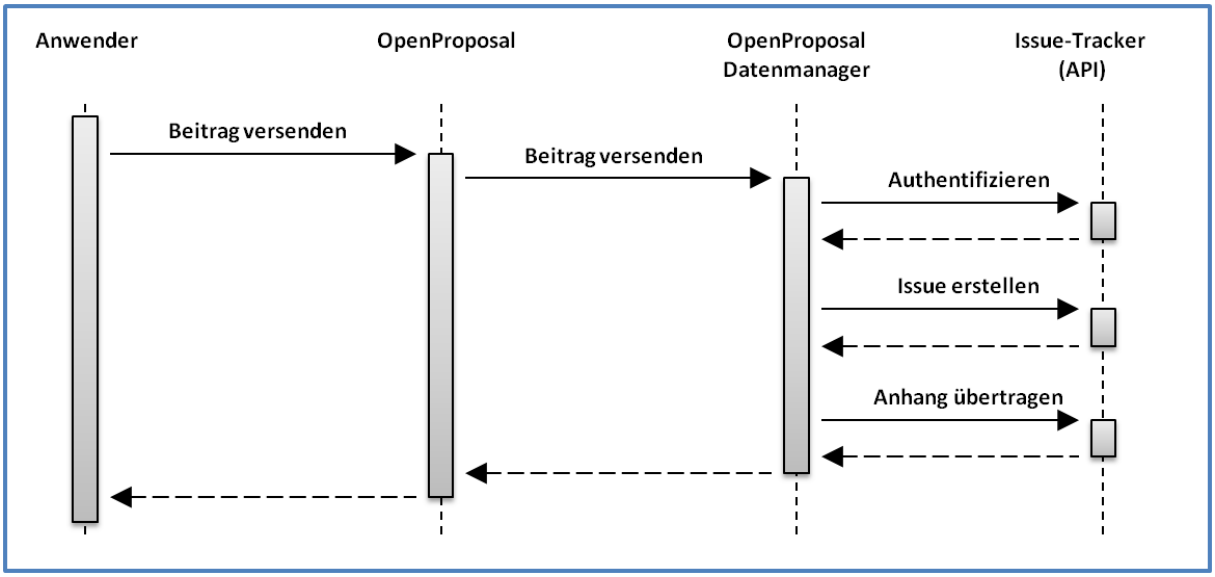

Abbildung 7.2: Sequenzdiagramm „Senden an ein Issue-Tracker“ mit OpenProposal

Für die Anbindung von Email-Systemen werden die Email-Funktionalitäten des Microsoft .NET Frameworks der Klassenbibliothek „System.Net.Mail“ verwendet. Zur Integration sind die URL des Email-Postausgangsserver, das Email-Konto des Anwenders und des Empfängers und optional - je nach Sicherheitseinstellungen des Email-Servers - die Anmeldedaten des Anwenders erforderlich. Anhand dieser Daten kann jedes Email-System jedes beliebigen Herstellers angebunden werden. OpenProposal erstellt beim Absenden eine Email mit dem Anwenderbeitrag als Anhang und versendet diesen an den Empfänger. In Abbildung 7.3 wird das Absenden eines Anwenderbeitrags per Email in einem Sequenzdiagramm illustriert.

[21] Intland Codebeamer, http://www.intland.com/, abgerufen am 16.03.2009

[22] Polarion ALM, http://www.polarion.com, abgerufen am 16.03.2009

[23] GForge, http://gforge.org/, abgerufen am 16.03.2009

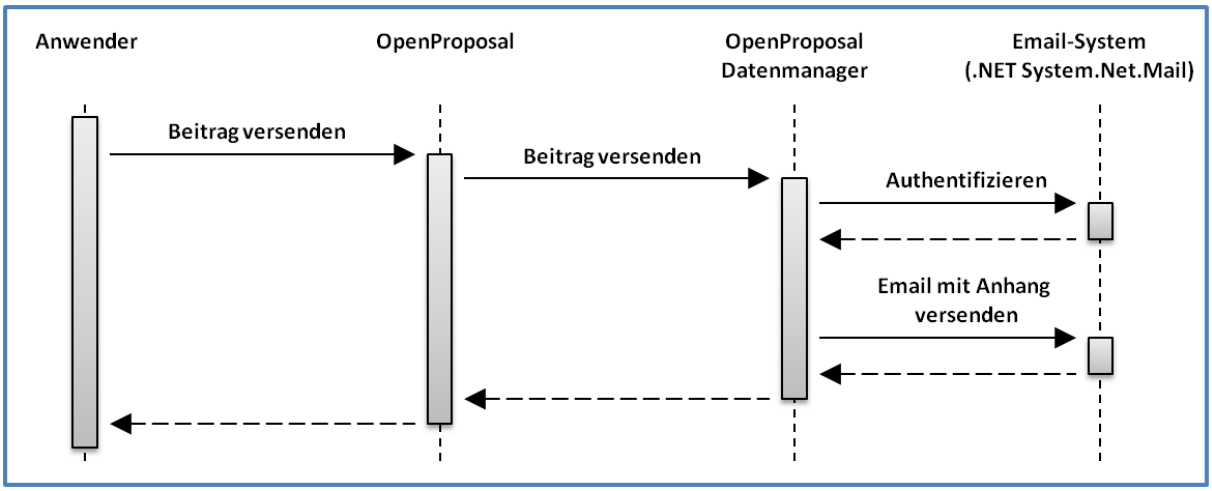

Abbildung 7.3: Sequenzdiagramm „Senden per Email“ mit OpenProposal

Die Anbindung von externen Webservices wird über die Regeldatenbank konfiguriert. Über Regeln können Dienste der externen Webservices aufgerufen und Daten übermittelt werden. Hierzu muss der externe Webservice Funktionen zum Empfangen des Verbesserungsvorschlages zur Verfügung stellen.

7.2 Datenmodell von OpenProposal

Der Entwicklung von OpenProposal liegt ein Datenmodell zugrunde, welches die Abbildung der Datenobjekte in OpenProposal beschreibt und eine einheitliche und konsistente Implementierung der Komponenten sicherstellt. Zur Modellierung der Datenobjekte wurden UML Entity-Relationship-Diagramme (ER-Diagramme) verwendet.

Im Folgenden wird die in OpenProposal verwendete Struktur für einen Anwenderbeitrag vorgestellt, wobei ausgewählte Aspekte genauer erläutert werden. Zu diesen gehören die Generierung des Titels und die Kurzbeschreibung eines Verbesserungsvorschlags, die Struktur der Systemparameter eines Verbesserungsvorschlags und die abstrahierte Struktur eines Issue-Trackers sowie eines Email-Systems.

7.2.1 Struktur eines Anwenderbeitrags von OpenProposal

Das zentrale Datenobjekt von OpenProposal ist der Anwenderbeitrag. Wie Abbildung 7.4 darstellt, wird der Anwenderbeitrag mit OpenProposal erstellt und vom OpenProposal Datenmanager optional zur weiteren Verarbeitung versendet. Der Anwenderbeitrag wird dann entweder als Issue im Issue-Tracker, als Email im Email-System gespeichert oder per Dienstaufruf an einen Webservice übermittelt.

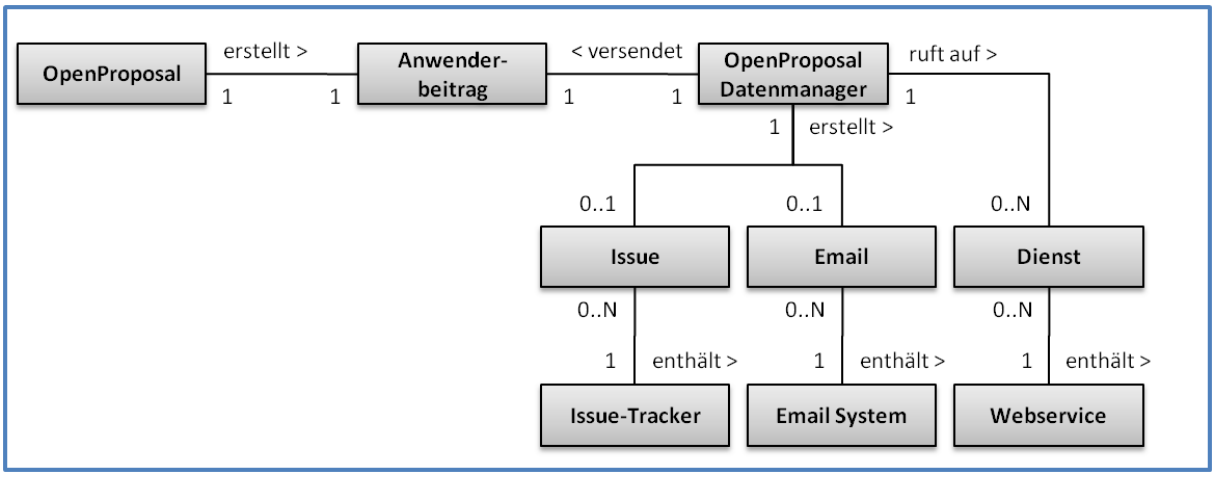

Abbildung 7.4: Zusammenspiel der Komponenten und der Datenobjekte in OpenProposal

In Abbildung 7.5 werden die Bestandteile eines Anwenderbeitrags von OpenProposal aufgeführt. Ein Anwenderbeitrag besteht demnach aus einem Satz Metadaten, einem Original-Bildschirmfoto und den vom Anwender erstellten Annotationen. Außerdem enthält der Anwenderbeitrag zusätzlich ein annotiertes Bildschirmfoto als Vorschauelement, das aus dem Bildschirmfoto und den Annotationen zusammengesetzt wird. Die vom Anwender mithilfe der Schaltfläche „Bildschirmfoto anpassen“ ausgeschnittenen Bereiche werden in der Bilddatei des Original-Bildschirmfotos sowie des annotierten Bildschirmfotos weiß überdeckt, so dass diese permanent ausgeblendet bleiben.

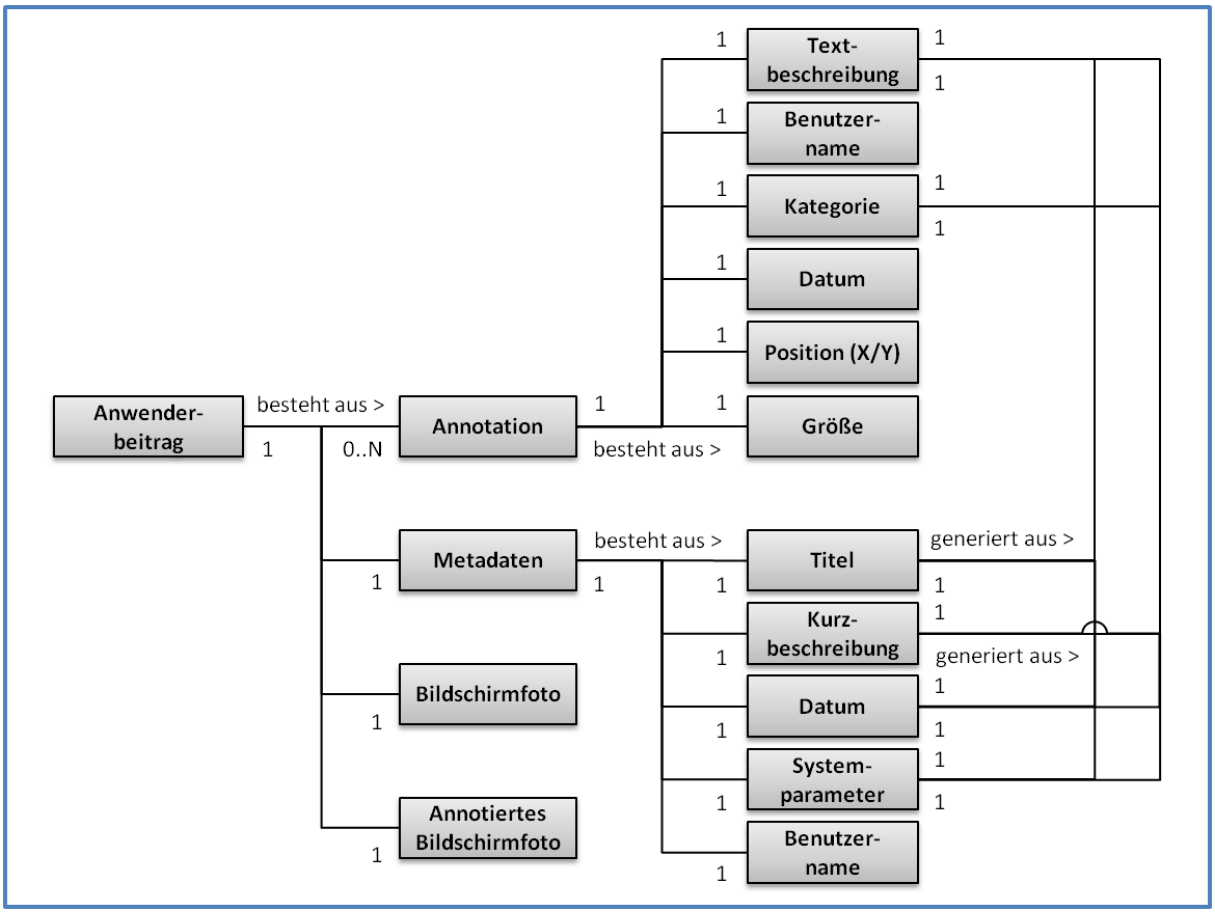

Abbildung 7.5: Datenmodell eines Anwenderbeitrags von OpenProposal

Eine Annotation besteht aus einer Textbeschreibung als zusätzlicher textlicher Erläuterung, dem Benutzername des Erstellers, der Kategorie der Annotation, dem Datum der Erstellung der Annotation, der grafischen Position auf dem Bildschirmfoto und der Größe des Markierungsbereichs. Ein Satz von Metadaten besteht aus einem generierten Titel und einer Kurzbeschreibung, dem Datum der Speicherung des Vorschlags, weiteren Systemparametern zum Anwendungssystem und dem Benutzernamen desjenigen Anwenders, der den Vorschlag erstellt hat. Zu jedem Vorschlag und zu jeder einzelnen Annotation wird der Name des Erstellers gespeichert. So kann später auch zwischen der Person, die den Anwenderbeitrag erstellt, und der Person, die bei der Überarbeitung zu einem späteren Zeitpunkt Annotationen ergänzt bzw. verändert hat, unterschieden werden.

Die Umsetzung des Datenmodells in XML ist in Abbildung 7.6 beispielhaft dokumentiert. Aufgrund des Umfangs der Datei wurde die Darstellung an mehreren Stellen gekürzt und die gekürzten Bereiche mit „...“ markiert. In der XML-Datei wird das Bildschirmfoto im JPEG-Format (als Base64 kodierte Zeichenfolge) gespeichert, um den Platzbedarf zu reduzieren. Alle anderen Daten werden nicht kodiert, da keine Notwendigkeit besteht. Dies hat zudem den positiven Nebeneffekt, dass die XML-Datei mit jedem beliebigen Textverarbeitungsprogramm betrachtet werden kann.

```
<?xml version="1.0" encoding="utf-8"?>
<specification>
  <metadata>
    <date>12.06.2009 20:48:58</date>
    <userName>Asarnusch</userName>
    <specificationTitle>Kein Titel</specificationTitle>
    <specificationDescription>Keine Beschreibung</specificationDescription>
    …
    <computer>
      <computerName>MEKONG</computerName>
      …
    </computer>
    <os>
      <platform>Win32NT</platform>
    </os>
  </metadata>
  <annotations>
    <annotationObject type="RECTANGLE_COMMEN">
      <guid>81316d77-bc8a-4bdb-b726-80372e354de5</guid>
      <authorName>Asarnusch</authorName>
      <authoredDate>12.06.2009, 20:49</authoredDate>
      <position x="356" y="466" />
      <size width="468" height="581" />
      <description>Hierzu bräuchte ich noch mehr Informationen!</description>
    </annotationObject>
  </annotations>
  <images>
    <blankSnapshot>
      <size>162344</size>
      <data encoding="Base64">/9j/4AAQSkZJR … </data>
    </blankSnapshot>
    <annotatedSnapshot>
      <size>167365</size>
      <data encoding="Base64">/9j/4AAQSkZJR …</data>
    </annotatedSnapshot>
  </images>
</specification>
```

Abbildung 7.6: Beispiel einer XML-Datei eines Anwenderbeitrags von OpenProposal

7.2.2 Generator für Titel und Kurzbeschreibung eines Anwenderbeitrags

Der Titel und die Kurzbeschreibung eines Anwenderbeitrags werden automatisch aus den Textbeschreibungen und Kategorien der Annotationen sowie den Systemparametern generiert. Der Algorithmus zur Textgenerierung von Titel und Kurzbeschreibung wird in Abbildung 7.7 als Pseudocode erläutert. Ziel der Titelgenerierung ist es, einen Titel zu finden, der für Moderator, Entscheider und Entwickler zur eindeutigen Identifikation eines Vorschlags ausreicht und ihnen einen schnellen Überblick über den Inhalt des Vorschlags gibt. Hierzu werden die Kategorie und die Textbeschreibung der Annotationen sowie der Name der annotierten Software als Titel verwendet. Falls mehrere Annotationen in einem Anwenderbeitrag erstellt wurden, werden die ersten fünf Wörter der Textbeschreibung verwendet, um den Titel dadurch kurz zu halten. In der Kurzbeschreibung ist eine solche Limitation nicht notwendig, so dass diese eine vollständige textliche Darstellung der Annotationen enthält. Dadurch soll es möglich sein, sich bereits vor dem Betrachten des annotierten Bildschirmfotos einen ersten Überblick über den Anwenderbeitrag verschaffen zu können.

```
Wenn // falls eine Annotation erstellt wurde, generiere den Titel des Verbesserungsvorschlags
                $Verbesserungsvorschlag.Annotation[] existiert UND
                $Verbesserungsvorschlag.Annotation[].Anzahl = 1
Dann
                $Anwenderbeitrag.Metadaten.Titel =
                  $Anwenderbeitrag.Systemparameter.AnnotierteSoftware[0].Name +
                  $Anwenderbeitrag.Annotation[0].Kategorie +
                  $Anwenderbeitrag.Annotation[0].Textbeschreibung;

Sonst Wenn // falls mehrere Annotationen erstellt wurden, generiere den Titel des Anwenderbeitrags
                $Anwenderbeitrag.Annotation[] existiert UND
                $Anwenderbeitrag.Annotation[].Anzahl > 1
Dann
                Für (X = 0 bis X = Anwenderbeitrag.Annotation[].Anzahl)
                   $Anwenderbeitrag.Metadaten.Titel =
                     $Anwenderbeitrag.Systemparameter.AnnotierteSoftware[X].Name +
                     $Anwenderbeitrag.Annotation[X].Kategorie +
                     Zeichensatz ( „Die ersten fünf Wörter“ von
                     $Anwenderbeitrag.Annotation[X].Textbeschreibung);

Wenn // falls Annotationen erstellt wurden, generiere die Kurzbeschreibung des Anwenderbeitrags
                $Anwenderbeitrag.Annotation[] existiert
Dann
                Für (X = 0 bis X = Anwenderbeitrag.Annotation[].Anzahl)
                  $ Anwenderbeitrag.Metadaten.Kurzbeschreibung =
                     $Anwenderbeitrag.Systemparameter.AnnotierteSoftware[X].Name +
                     $Anwenderbeitrag.Systemparameter.AnnotierteSoftware[X].Version +
                     $Anwenderbeitrag.Annotation[X].Kategorie +
                     $Anwenderbeitrag.Annotation[X].Textbeschreibung);

Sonst // falls keine Annotation existiert, generiere Titel und Kurzbeschreibung des Anwenderbeitrags
                $Anwenderbeitrag.Metadaten.Titel =
                     $Anwenderbeitrag.Systemparameter.Computer.Name +
                     $Anwenderbeitrag.Systemparameter.Betriebssystem.Name +
                     $Anwenderbeitrag.Metadaten.Datum;
                $ Anwenderbeitrag.Metadaten.Kurzbeschreibung =
                     $Anwenderbeitrag.Systemparameter.Betriebssystem.Name +
                     $Anwenderbeitrag.Systemparameter.Betriebssystem.Version +
                     $Anwenderbeitrag.Systemparameter.Betriebssystem.GeöffneteProgramme;
```

Abbildung 7.7: Pseudocode zur Generierung von Titel und Kurzbeschreibung

7.2.3 Struktur der Systemparameter eines Anwenderbeitrags

Abbildung 7.8 beschreibt die Bestandteile der Systemparameter eines Anwenderbeitrags. Diese beziehen sich auf den vom Anwender verwendeten Computer, das darauf installierte Betriebssystem und auf die annotierten Software-Produkte. Die wichtigsten Informationen zum Computer sind der Computername, die zugewiesene IP-Adresse, die Merkmale des Prozessors bzw. der Prozessoren und des Arbeitsspeichers. Bezüglich des Betriebssystems sind Name, Version und der Anmeldename des Anwenders von Relevanz. Außerdem können Informationen zu installierten Laufzeitumgebungen und Treibern erfasst werden. Um festzustellen, welche Software annotiert wurde, sind außerdem die geöffneten Software-Produkte zu identifizieren, um über deren Position und Größe festzustellen, ob diese Software mit einer Annotation versehen wurde.

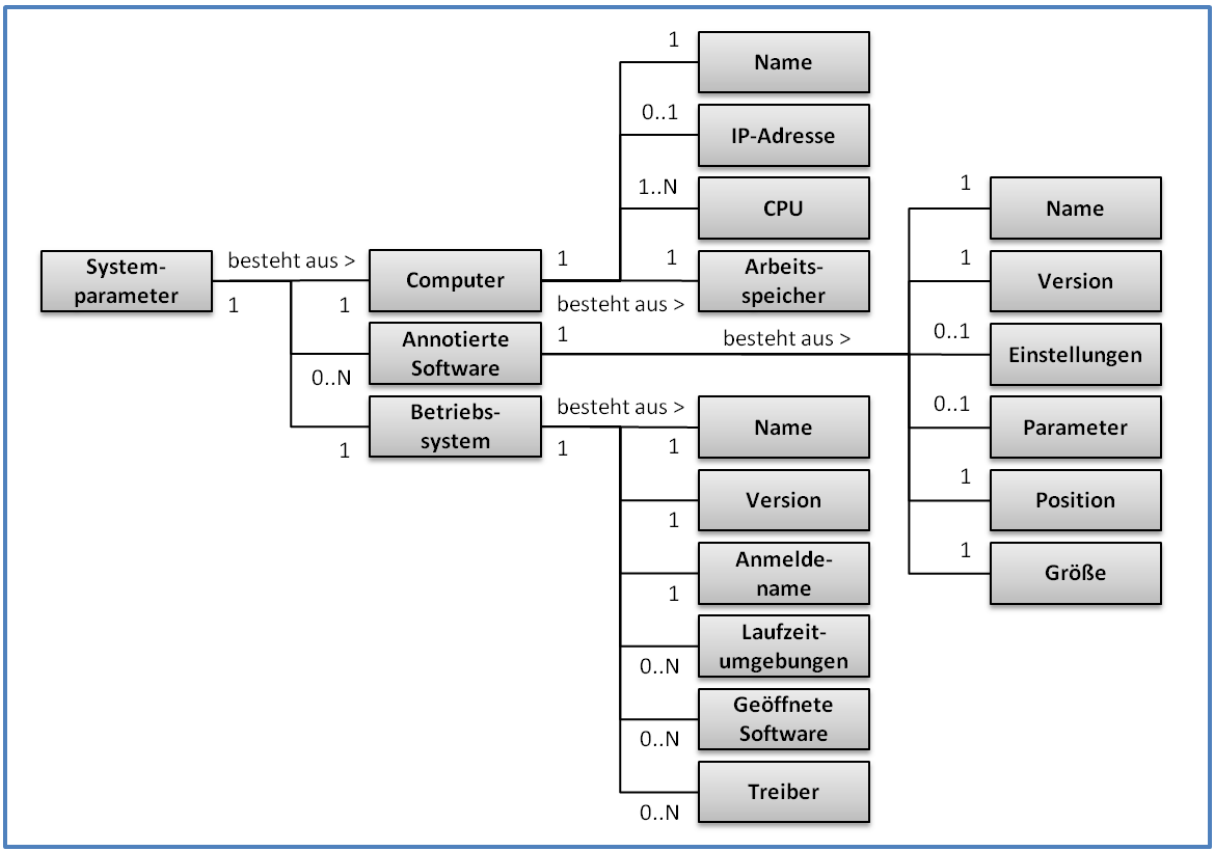

Abbildung 7.8: Datenmodell der Systemparameter von OpenProposal

Zu jeder annotierten Software wird deren Name, Version, Position und Größe gespeichert. Falls möglich, werden weitergehende Parameter (z.B. geöffnete Anwendungsfenster, geöffnete URL, geöffnete Datenbank, geöffnete Datenobjekt, etc.) und die Einstellungen der Software erfasst. Allerdings sind diese Daten spezifisch für jede Software vorab in OpenProposal zu implementieren, da sich jede Software in ihrer Datenstruktur unterscheidet. Außerdem muss in der Software implementiert werden, welche Datenobjekte über das Betriebssystem bzw. OpenProposal ausgelesen werden können.

7.2.4 Struktur von Issues und Emails

Um einen Anwenderbeitrag als Issue in einem Issue-Tracker zu speichern oder per Email zu versenden, ist eine Übersetzung der Struktur des eines Anwenderbeitrags in die Struktur eines Issues bzw.

einer Email notwendig. Hierzu werden im Folgenden eine Abstraktion der Struktur eines Issues sowie einer Email erarbeitet und die Abbildung eines Anwenderbeitrags von OpenProposal auf einen Issue oder eine Email erläutert. Abbildung 7.9 stellt die erarbeiteten Abstraktionen eines Issues und einer Email grafisch dar.

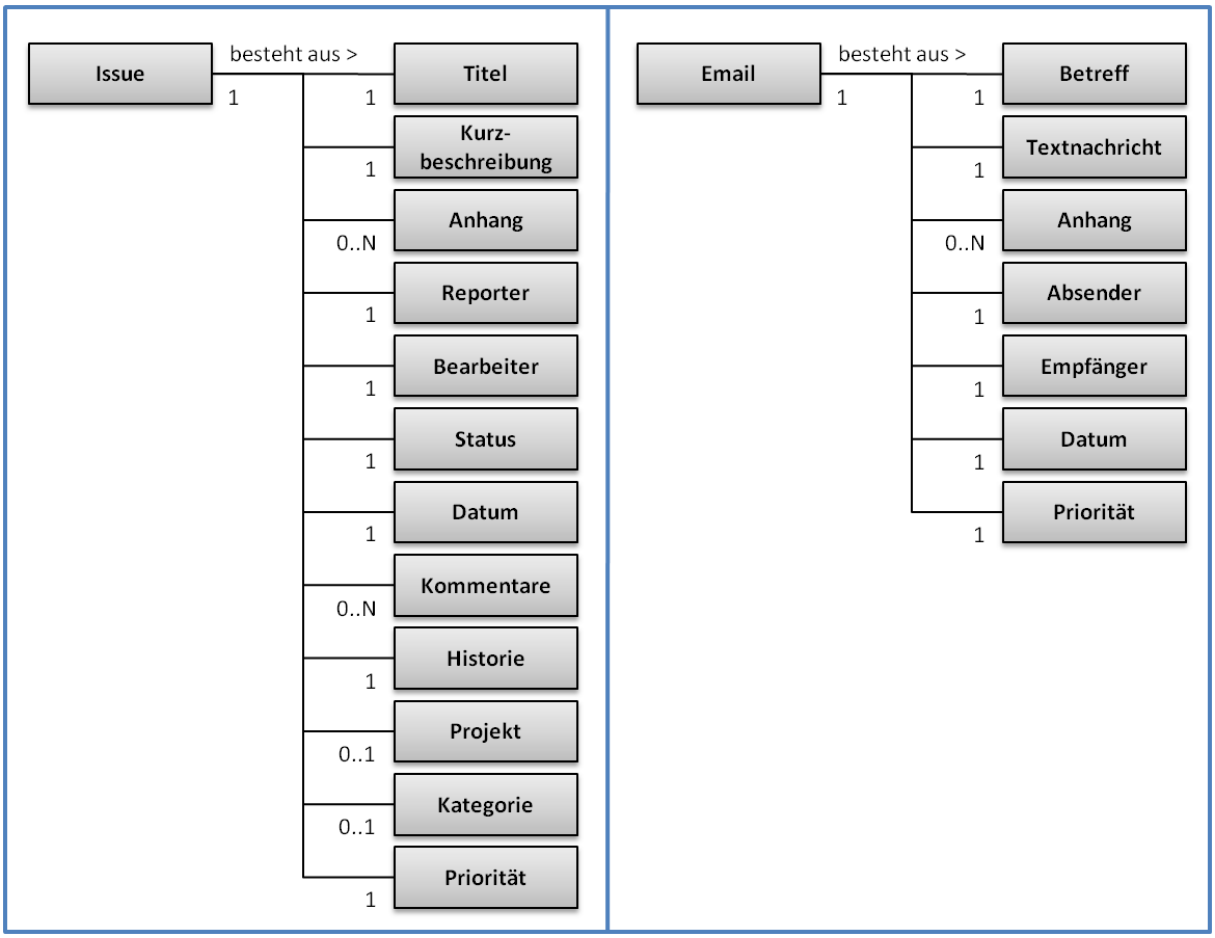

Abbildung 7.9: Datenmodell von einem Issue (l.) und einer Email (r.)

Bei einer Email ist der global standardisierte Kodierstandard „Multipurpose Internet Mail Extensions“ (MIME)[24] vorhanden, der für die Abstraktion der wichtigsten Bestandteile herangezogen werden kann. Dieser besteht im Allgemeinen aus einem Betreff, einer Textnachricht, einem Absender, einem Empfänger, dem Datum des Versands und der Priorität der Email. Zudem kann eine Email einen oder mehrere Anhänge besitzen.

Bei Issue-Trackern existiert hingegen keine standardisierte Datenstruktur, sodass hierzu eine Datenstruktur aus den auf dem Markt befindlichen Issue-Trackern JIRA von Atlassian[Fehler! Textmarke nicht definiert.], Polarion ALM von Polarion[22] und Codebeamer von Intland[21] sowie aus den Open-Source-Software Issue Trackern GForge[23], Track+[25] und SourceForge.Net[26] abstrahiert werden muss. Für OpenProposal wurden die für die Speicherung eines Anwenderbeitrags relevanten Datenobjekte berücksichtigt.

---

[24] RFC 2387 MIME, http://tools.ietf.org/html/rfc2387, abgerufen am 07.03.2009

[25] Track+, http://sourceforge.net/projects/trackplus/, abgerufen am 16.03.2009

Jeder Issue besteht aus einem Titel, einer Kurzbeschreibung, einem Reporter (dem Autor des Issue), einem aktuellen Bearbeiter des Issue, dem Datum seiner Erstellung, einer Historie der Bearbeitung, und einer zugeordneten Priorität bezüglich dessen Bearbeitung. Außerdem können an einen Issue Dateien angehängt und Kommentare hinzugefügt werden. Im Laufe der Bearbeitung können einem Issue zudem ein Projekt und eine Projektkategorie zugeordnet werden.

Issue Datenobjekt	**Email Datenobjekt**	**Datenobjekt-Zuordnung zum Anwenderbeitrag**
Titel	Betreff	Anwenderbeitrag.Metadaten.Titel
Kurzbeschreibung	Textnachricht	Anwenderbeitrag.Metadaten.Kurzbeschreibung
Anhang	Anhang	Anwenderbeitrag.Annotiertes Bildschirmfoto und Verbesserungsvorschlag (als XML-Datei)
Reporter	Absender	Anwenderbeitrag.Metadaten.Benutzername
Bearbeiter	-	(nicht abgebildet, automatisch durch Issue-Tracker)
-	Empfänger	(Standard-Zeichensatz)
Status	-	(nicht abgebildet, automatisch durch Issue-Tracker)
Datum	Datum	Anwenderbeitrag.Metadaten.Datum
Kommentare	-	(nicht abgebildet)
Historie	-	(nicht abgebildet)
Projekt	-	(optional, Metadaten oder Standard-Zeichensatz)
Kategorie	-	(optional, Metadaten oder Standard-Zeichensatz)
Priorität	Wichtigkeit	(optional, Metadaten oder Standard-Zeichensatz)

Tabelle 7.1: Datenobjekt-Zuordnung von Issue und Email in OpenProposal

Die Zuordnung der Datenobjekte eines Anwenderbeitrags zu einem Issue gestaltet sich folgendermaßen: Der Titel und die Kurzbeschreibung generieren sich aus den Metadaten des Anwenderbeitrags. Das annotierte Bildschirmfoto und der gesamte Anwenderbeitrag werden als XML-Datei im Anhang hinzugefügt. Der Reporter eines Issue besteht aus dem Benutzernamen des OpenProposal Benutzers. Als Datum wird das Erstellungsdatum des Anwenderbeitrags gewählt. Optional können die Projektzuordnung, die Kategorie und die Priorität gewählt werden. Hierzu werden in der Regel-Datenbank des Datenmanagers Regeln definiert, anhand derer dem Issue ein Projekt, eine Kategorie bzw. eine Priorität zugeordnet werden. Der Bearbeiter und der Status werden beim Eintragen eines Issue vom Issue-Tracker automatisch ausgewählt. Eine Abbildung auf die Datenobjekte Kommentare und Historie erfolgt nicht, da diese erst im Laufe der Bearbeitung erstellt werden und somit für

[26] SourceForge SourceForge.Net, http://sourceforge.net/, abgerufen am 16.03.2009

OpenProposal keine Rolle spielen. Die Zuordnung der Datenobjekte eines Anwenderbeitrags zu einer Email erfolgt in ähnlicher Form. Eine Übersicht der gewählten Zuordnung gibt Tabelle 7.1 wieder.

7.3 Bedienoberfläche von OpenProposal

Die Bedienoberfläche von OpenProposal stellt die grafische Repräsentation des Annotationssystems dar und beinhaltet die in den vorherigen Kapiteln erarbeiteten Konzepte. Bei der Gestaltung der Bedienoberfläche wurden die allgemeinen software-ergonomischen Anforderungen berücksichtigt, die sowohl die Effektivität bei der Aufgabenlösung, die Effizienz bei der Werkzeughandhabung als auch die Zufriedenheit des Benutzers gewährleisten sollen.

Die Gestaltung der Bedienoberfläche erfolgte im regelmäßigen Dialog mit Anwendern im Rahmen häufig stattfindender Benutzertests. Bereits zu einem frühen Stadium wurde die Optik eines Notizhefts zur Darstellung des Fensters gewählt, da diese Form der Darstellung in keiner bekannten Anwendung Verwendung findet und sich somit bei jeder beliebigen Software deutlich von der zu annotierenden Benutzeroberfläche abhebt. Im weiteren Verlauf wurde die Optik überarbeitet, vereinfacht und auf das Wesentliche reduziert. Es wurden symbolische Darstellungen bevorzugt und weitestgehend auf textliche Beschreibungen verzichtet, um einen übersichtlichen Eindruck zu vermitteln. Bei der Gestaltung von Schaltflächen wurde auf eine konsistente und selbsterklärende Darstellung geachtet, um dem Anwender eine einfache und einprägsame Bedienung anzubieten.

Abbildung 7.10 zeigt den Start- und Hauptbildschirm von OpenProposal. Der Notizblock dient als Darstellung des Fensters und bildet den Hintergrund für die Schaltflächen. Das Fenster weist das übliche Verhalten eines Fensters auf und lässt sich minimieren und schließen. Solange es nicht minimiert wird, bleibt das Fenster stets im Vordergrund, auch wenn andere Anwendungen aktiviert werden. Dadurch soll sichergestellt werden, dass nicht zwischen den Anwendungen und OpenProposal gewechselt werden muss. Beim Starten von OpenProposal wird der Annotationsmodus angezeigt, wie in Abbildung 7.10 dargestellt, so dass direkt mit dem Annotieren begonnen werden kann.

Abbildung 7.10: Start- und Hauptbildschirm von OpenProposal

Das Fenster ist in die fünf Hauptbereiche „Hauptmenü“ (1), „Fenstersteuerung“ (2), „Annotationswerkzeuge“ (3), „Versandsteuerung“ (4) und „Bildschirmfotosteuerung“ (5) gegliedert. Das Hauptmenü beinhaltet alle Befehle, die nur sekundär von Bedeutung und nicht primär zum Annotieren notwendig sind. Über die Fenstersteuerung kann OpenProposal geschlossen, minimiert und aktiviert bzw. deaktiviert werden. Wenn OpenProposal in den Status „Aktiv“ gesetzt wird, wird die aktuelle Darstellung des Anwendungsbildschirms in einem Bildschirmfoto „eingefroren“. Dabei wird ein

„grauer Schleier“ über den Bildschirm gelegt, sodass der Anwender sich dieser Tatsache bewusst bleibt. Er hat nun die Möglichkeit auf diesem Bildschirmfoto zu annotieren. Wird OpenProposal deaktiviert, werden die Annotationselemente ausgeblendet und der Anwendungsbildschirm freigegeben, so dass der Anwender seine Anwendungen benutzen kann. OpenProposal bleibt dabei immer im Vordergrund und steht somit sofort wieder bereit, wenn man das Annotationssystem wieder aktivieren möchte.

Die Annotationswerkzeuge bieten dem Anwender die Möglichkeit, Annotationen auf der Anwendungsoberfläche zu platzieren. Über die Versandsteuerung kann der annotierte Anwenderbeitrag gespeichert bzw. versendet werden. Mit der Bildschirmfotosteuerung kann der Anwender den Sichtbereich des Bildschirmfotos anpassen und damit persönliche Informationen auf dem Bildschirmfoto ausblenden. Als Annotationswerkzeuge stehen in Anlehnung an das OpenProposal Konzept in Abbildung 6.14 folgende Möglichkeiten zur Verfügung (vgl. Abbildung 7.11): Hinzufügen einer Funktion (1), Verschieben einer Funktion (2), Entfernen einer Funktion (3), Fehler in einer Funktion (4), Verbessern einer Funktion (5), Frage zu einer Funktion (6), Lob zu einer Funktion (7) und Kommentar zu einer Funktion (8).

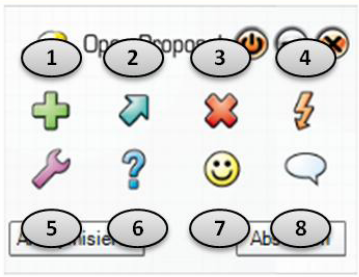

Abbildung 7.11: Annotationswerkzeuge von OpenProposal im Annotationsmodus

Abbildung 7.12: Beispiel „Kommentar zu einer Funktion“ mit OpenProposal

Die Benutzung der Annotationswerkzeuge erfolgt nach einem einfachen Prinzip: Ein Annotationswerkzeug wird durch Anklicken der entsprechenden Schaltfläche aktiviert. Anschließend kann der Benutzer auf dem Bildschirmfoto der Anwenderoberfläche einen Bereich markieren und im geöffneten Textfeld einen ergänzenden Text eingeben. Abbildung 7.12 illustriert anhand eines Beispiels die generelle Funktionsweise zur Platzierung einer Annotation. In (1) wird Annotationswerkzeug „Kommentieren einer Funktion“ ausgewählt, dann wird auf der Anwendungsoberfläche mit einer rechteckigen Markierung (2) das Objekt der Verbesserung ausgewählt und abschließend ein Text zur Annotation (3) eingegeben.

Abbildung 7.13 demonstriert anhand von Anwendungsbeispielen die Funktionsweise der unterschiedlichen Annotationswerkzeuge von OpenProposal. Objekt des Vorschlags stellt die Software „Rechner“, eine Variante eines Taschenrechners in Microsoft Windows Vista, dar. Bei der ersten Annotation (1) wird ein neuer Button vorgeschlagen. Unter (2) wird gewünscht, dass die markierte Schaltfläche nach oben verschoben werden soll. In (3) wird eine Schaltfläche als überflüssig beschrieben. Bei (4) meldet der Anwender einen Fehler in der Bezeichnung einer Schaltfläche. Mit der fünften Annotation (5) soll die Verständlichkeit einer Schaltfläche durch eine bessere Begrifflichkeit erhöht werden. Mit (6) gibt der Anwender einen allgemeinen Kommentar ab, dass er mehrere Schaltflächen zusammenfassen würde. Dass ihm die Anwendung gut gefällt, gibt er in (7) wieder. Unter (8) stellt der Anwender eine Frage zu mehreren Schaltflächen.

Das Hauptmenü kann über eine Schaltfläche im OpenProposal-Logo geöffnet werden (siehe Abbildung 7.14) und bietet dem Anwender die Möglichkeit, zwischen dem Annotations- und dem Konfigurationsmodus zu wechseln. Außerdem hat der Anwender die Wahl, einen neuen Vorschlag zu beginnen, einen bereits abgesendeten Vorschlag zu öffnen bzw. zu bearbeiten oder ein Tutorial bzw. eine Einführung zu starten. Darüber hinaus kann der Anwender das von OpenProposal erstellte Bildschirmfoto über die Funktion „Bildschirmfoto neu erstellen“ erneuern. Diese Funktion ist sinnvoll, wenn beim Aktivieren von OpenProposal nicht automatisch ein neues Bildschirmfoto erstellt werden soll. Wenn diese Einstellung gesetzt ist, kann der Anwender das Bildschirmfoto über diese Funktion manuell aktualisieren lassen.

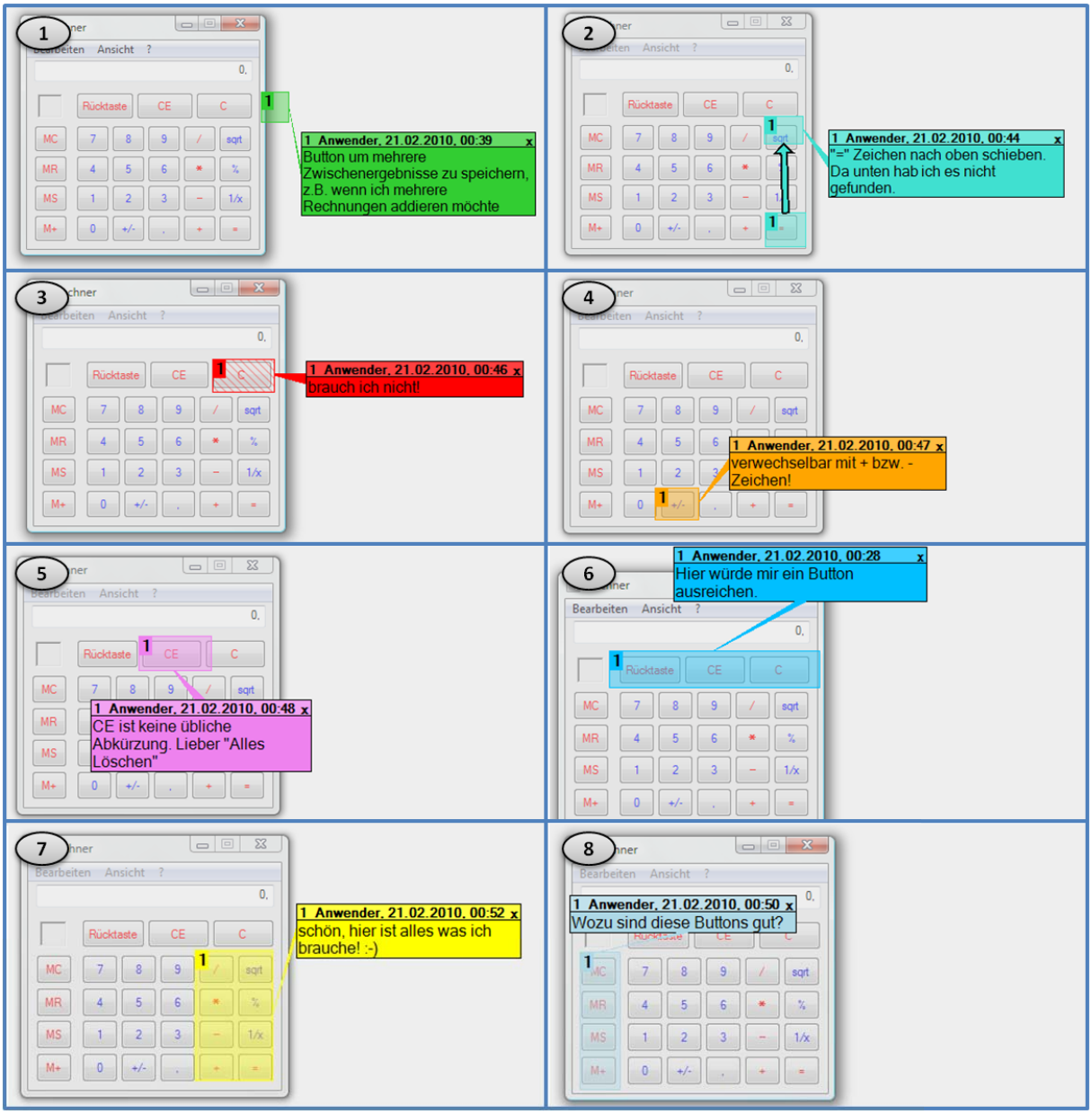

Abbildung 7.13: Beispiele für die Annotationswerkzeuge von OpenProposal

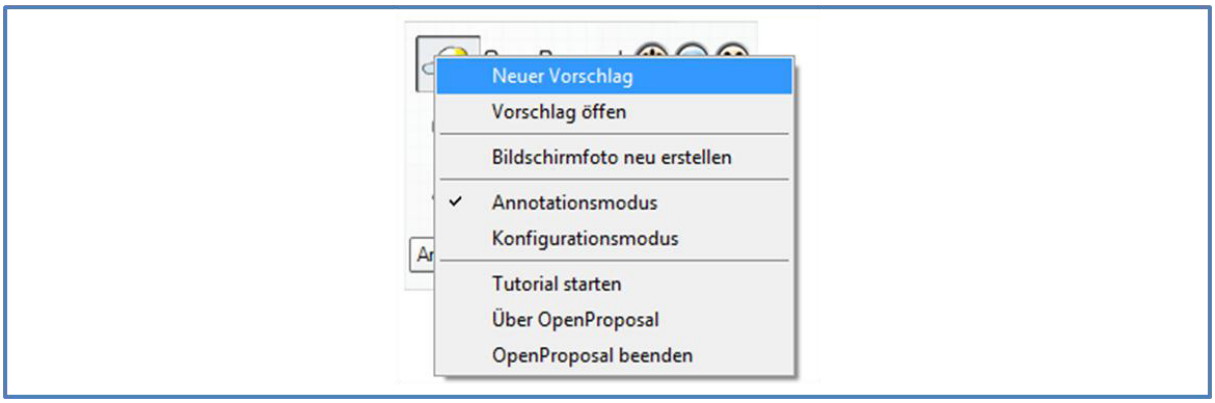

Abbildung 7.14: Hauptmenü von OpenProposal

Abbildung 7.15 illustriert den Konfigurationsmodus von OpenProposal, der über das Hauptmenu aktiviert wird. Dieser erlaubt die Anpassung der Einstellungen in OpenProposal. Neben Benutzername, Sprache und anderen allgemeinen Einstellungen (1) können an dieser Stelle die Darstellung (2) und die Art und das Ziel des Absendens (3) konfiguriert werden.

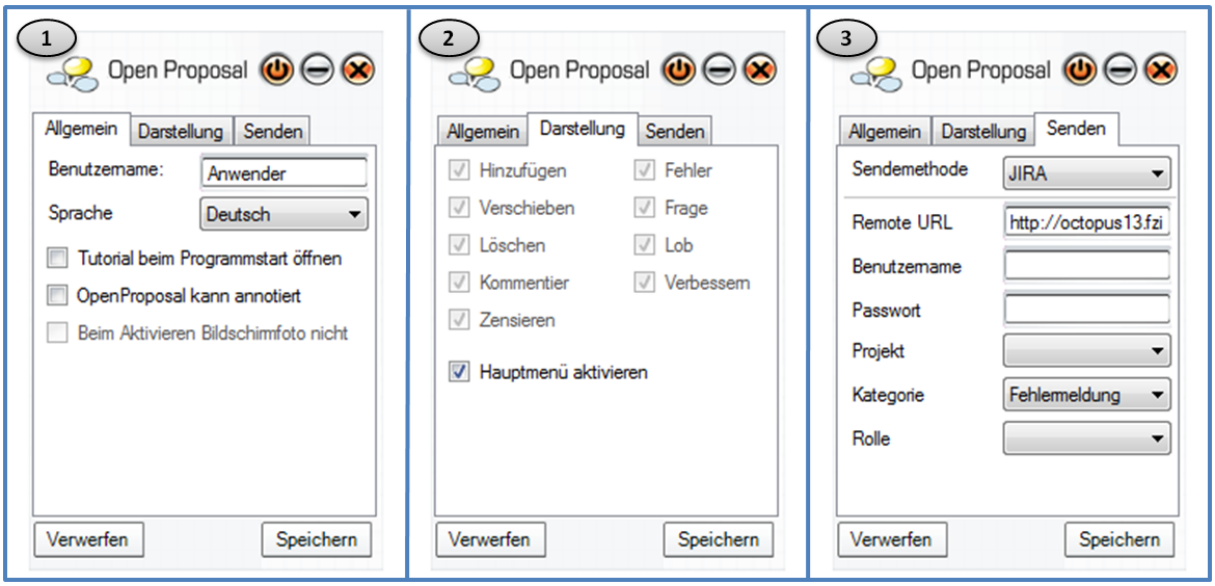

Abbildung 7.15: Konfigurationsmodus: Allgemein (1), Darstellung (2) und Senden (3)

Beim Absenden des Anwenderbeitrags wird der Anwender über den Verlauf des Absendens und über den Speicherort des Anwenderbeitrags informiert. Abbildung 7.16 illustriert diesen Vorgang beispielhaft für den Versand eines Anwenderbeitrags als Email.

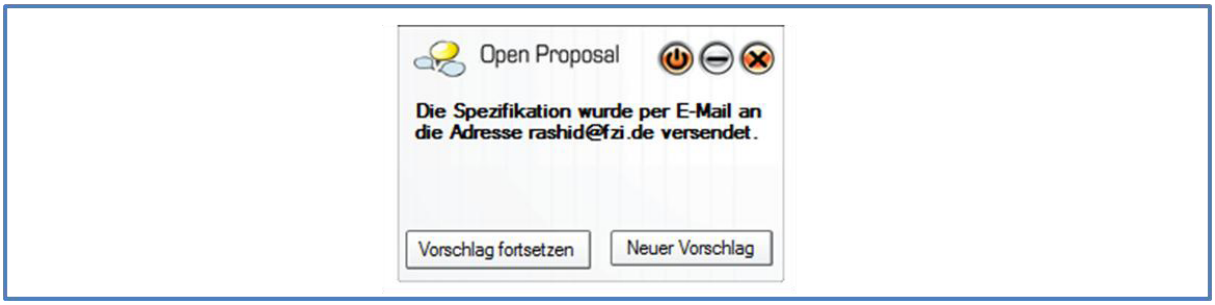

Abbildung 7.16: Dialog für Absenden eines Anwenderbeitrags mit OpenProposal

Um einen abgesendeten Anwenderbeitrag zu betrachten, kann der Anwenderbeitrags in OpenProposal geöffnet oder die Bilddatei des annotierten Bildschirmfotos mit jedem beliebigen Bildanzeigeprogramm betrachtet werden. Abbildung 7.17 zeigt anhand eines Beispiels, wie ein Anwenderbeitrag von OpenProposal in einem Issue-Tracker geöffnet werden kann. Die Abbildung zeigt den webbasierten Issue-Tracker von OpenProposal.

In der Issue-Liste des Issue-Trackers (1) werden die einzelnen Issues mit den wichtigsten Daten (z.B. Datum, Titel, Status, Reporter, Bearbeiter, etc.) aufgelistet. In oberen Teil des Issue-Trackers sind die Navigation und eine Filterfunktion untergebracht. Über die Navigation kann zwischen der Übersichtsseite, der Liste aller Anwenderbeiträge und einer Maske zur Eingabe eines neuen Anwenderbeitrags bzw. Vorschlags gewechselt werden. Außerdem können Zusatzfunktionen wie z.B. das Änderungs-

protokoll, die Zusammenfassung, die Verwaltung und interne Nachrichten aufgerufen werden. Über die Filterfunktion können abhängig von der Einstellung des Filters die Auswahl der angezeigten Anwenderbeiträge vorgenommen werden. So können z.B. nur die Anwenderbeiträge angezeigt werden, die von einem bestimmten Anwender oder zu einem bestimmten Zeitpunkt erstellt wurden.

Durch Öffnen eines Issues (2) können weitere Informationen (z.B. Kurzbeschreibung, Historie, etc.) eingeholt werden. Zusätzlich kann das Bildschirmfoto (3) als Bilddatei direkt im Internetbrowser betrachtet oder als XML-Datei in OpenProposal geöffnet werden.

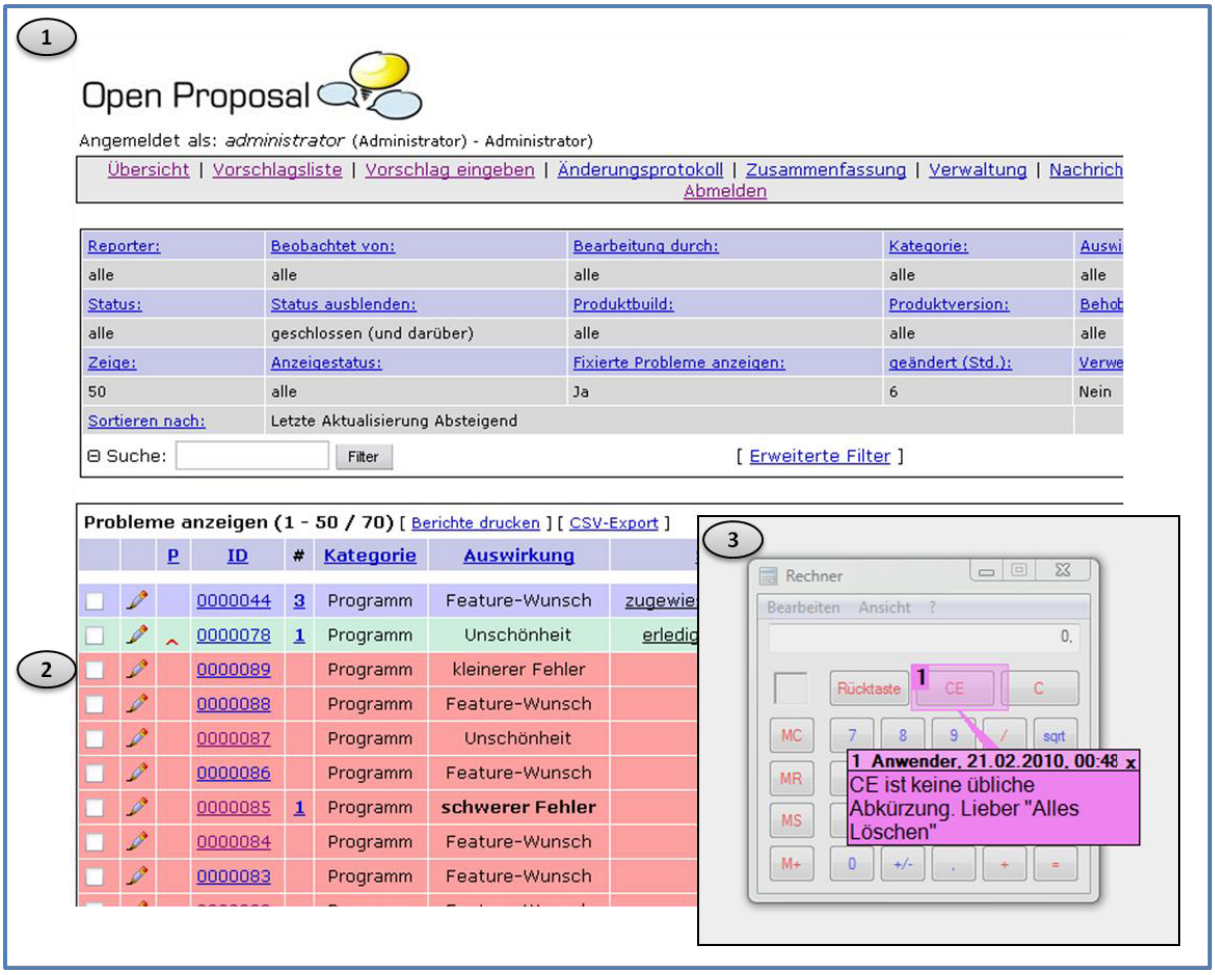

Abbildung 7.17: Beispiel der Sicht auf einen Verbesserungsvorschlag in einem Issue-Tracker

7.4 Bedienkonzept von OpenProposal

Um die grafische Benutzeroberfläche von OpenProposal zu komplettieren, erfordert eine Bedienoberfläche zusätzlich ein Bedienkonzept, um neben der grafischen Gestaltung auch die Mensch-Maschine-Interaktion mit dem Benutzer sowie das Verhalten des Werkzeuges an die gegebenen Anforderungen anzupassen. Das Bedienkonzept von OpenProposal orientiert sich an den zehn Gestaltungsprinzipien nach [Nielsen, 1994a; Nielsen, 1994b], ähnlich den „acht goldenen Regeln“ von Shneiderman und Plaisant [2005]. Ziel ist es, eine konsistente, erwartungskonforme und effiziente Bedienung des Werkzeuges nach den EN ISO 9241 Richtlinien zu bieten, wie es in Anforderung A1 zur Erfüllung der Gebrauchstauglichkeit gefordert wird:

Gestaltungsregel G1: Einfache und natürliche Dialoge („Simple and Natural Dialogue“)

Die Gestaltung der Benutzeroberfläche und der Interaktionsdialoge richten sich nach dem Motto „Weniger ist mehr“. So wurde darauf geachtet, dass nur die für die Aufgabe relevanten Informationen in den Dialogen angezeigt werden. Außerdem wurde auf eine sinnvolle Gruppierung der Schaltflächen geachtet und die Farbgebung zielgerichtet eingesetzt.

Gestaltungsregel G2: Sprich die Sprache des Anwenders („Speak the Users‘ Language“)

Die Dialoge und die Schaltflächen sind so gestaltet, dass der Anwender für ihre Bedienung keinen technologischen Hintergrund benötigt. Annotationen können mit einem einfachen „Klicken und Ziehen“ auf dem Bildschirm platziert und anschließend mit einer ergänzenden Beschreibung versehen werden – ähnlich dem Prinzip „Markieren und kommentieren“ wie es bei der papierbasierten Annotation verwendet wird. Technische Einschränkungen des Anwenders (z.B. maximale Anzahl an Annotationen, Begrenzung der Wortzahl, etc.) sind nicht vorhanden. Symbolik und Texte der Schaltflächen und Dialoge sind aus der Sicht der Anwender formuliert und sollen es dem Anwender erleichtern, die reale Welt auf die virtuelle Computer-Umgebung zu übertragen.

Gestaltungsregel G3: Minimiere die kognitive Beanspruchung („Minimize Users‘ Memory Load“)

Soweit es möglich war, wurden assistierende Funktionen in OpenProposal implementiert, um dem Anwender eine Erleichterung bei der Bedienung und der Formulierung von Feedback zu bieten. Bei der Bedienung ist hauptsächlich von Bedeutung, dass der Anwender bewusst ein Annotationswerkzeug auswählt und dieses auf dem Bildschirmfoto platziert. Alle weiteren Schritte werden automatisch angeboten. Beim Platzieren einer Annotation wird beispielsweise ein ergänzendes Beschreibungsfeld mit einem motivierenden Text eingefügt. Falls der Anwender dieses Feld nicht ausfüllt, wird dieses Feld beim Absenden des Vorschlages wieder ausgeblendet, um dem Anwender Arbeit abzunehmen und dem Moderator die Darstellung von leeren Textfeldern zu ersparen.

Gestaltungsregel G4: Konsistenz („Consistency“)

Um die Konsistenz in der Benutzerführung zu bewahren, wurde eine durchgängig eindeutige und klare Benutzerführung sichergestellt. Alle Schaltflächen verhalten sich zu jeder Zeit und in jedem Zustand gleich. Die Formulierung der Texte und die Darstellung der Grafiken wurden in demselben Stil gehalten. Die Platzierung der Annotationen und deren Bearbeitung (z.B. Verschieben, Größe ändern, etc.) verhalten sich bei jeder Kategorie ebenfalls gleich.

Gestaltungsregel G5: Rückmeldung („Feedback“)

Dem Anwender werden über jeden Schritt ausführliche Informationen angezeigt. Beispielsweise wird dem Anwender nach dem Absenden seines Vorschlages angezeigt, wie sein Vorschlag abgesendet wurde und welche weiteren Schritte sich nun anbieten würden. Außerdem reagiert OpenProposal auf die Aktionen des Anwenders. Wird OpenProposal aktiviert, wird die gesamte Bildschirmoberflä-

che leicht grau eingefärbt, so dass sich der Anwender bewusst wird, dass er nun annotieren kann. Wird ein Annotationswerkzeug ausgewählt, bleibt die Schaltfläche eingedrückt und der Mauszeiger nimmt die Form eines Fadenkreuzes an, um dadurch das Platzieren der Annotation anzuzeigen.

Gestaltungsregel G6: Klar erkennbare Abbruchmöglichkeiten („Clearly Marked Exits“)

Der Anwender kann das Programm jederzeit beenden. Über die Funktion „Neuer Vorschlag“ des Hauptmenüs kann der Anwender seinen Vorschlag erneut beginnen. Eine „Undo“-Funktion ist vorhanden, um fehlerhafte Eingaben jederzeit rückgängig machen zu können.

Gestaltungsregel G7: Abkürzungen („Shortcuts“)

Versierten Anwendern steht die Möglichkeit zur Verfügung, Tastaturkürzel verwenden. Annotationen können kopiert und wieder eingefügt werden(„copy and paste“). Schließlich werden an das Absenden eines Vorschlages keine Bedingungen gestellt, so dass ein Vorschlag auch aus einem Bildschirmfoto ohne jegliche Annotationen bestehen kann und dem Anwender die Eingabe unnötiger Annotationen dabei erspart wird.

Gestaltungsregel G8: Gute Fehlermeldungen („Good Error Messages“)

Sollten Fehler auftauchen, wird der Anwender unmittelbar darüber informiert und es werden ihm Vorgehensweisen zum Umgang mit diesem Fehler zur Auswahl gestellt. Bereits beim Starten von OpenProposal wird die aktuelle Konfiguration auf ihre Korrektheit geprüft und bei der Feststellung von Widersprüchen oder Fehlermöglichkeiten eine Warnung ausgesprochen. Besteht keine für das Absenden des Vorschlages erforderliche Netzwerkverbindung zum Kollaborationsportal, wird der Anwender bereits vor dem Erstellen des Vorschlages informiert und es wird ihm das Prüfen der Konfiguration empfohlen.

Gestaltungsregel G9: Fehler vermeiden („Prevent Errors“)

Vorhersehbare Fehlerquellen sollen vermieden bzw. auf ein Minimum reduzieret werden. Die größte Fehlerquelle stellt dabei die Konfiguration der Netzwerkverbindung zu einem Issue-Tracker oder einem Webservice mit damit einhergehendem Datenverlust dar. Sollte das Absenden fehlschlagen, wird daher stets eine Kopie des Vorschlages auf der Arbeitsstation erstellt, so dass die Vorschläge auch manuell (z.B. per USB-Stick, Email-Anhang) versendet werden können.

Gestaltungsregel G10: Hilfe und Dokumentation („Help and Documentation“)

Zur Installation und zur Einführung von OpenProposal existieren Dokumente, eine Webseite sowie Videos. Zudem hat der Anwender die Möglichkeit ein kurzes „In-Application“-Tutorial aufrufen, welches ihm die grundlegenden Eigenschaften von OpenProposal während der Verwendung erläutert und die Gelegenheit bietet, die Bedienmöglichkeiten anhand von Beispielen in wenigen Schritten selbst auszuprobieren.

7.5 Fazit zu Kapitel 7

Ziel von Kapitel 7 war die Beschreibung der Implementierung des OpenProposal Annotationssystems. Hierfür wurden die Systemarchitektur inklusive der einzelnen Komponenten, das Datenmodell, das Bedienkonzept und die Bedienoberfläche vorgestellt und im Detail erläutert. Bei der Entwicklung und Gestaltung des Annotationssystems wurden dabei die gängigen Normen der objektorientierten Programmierung, der UML-Modellierung und der Gebrauchstauglichkeit von Software herangezogen. Dabei wurde OpenProposal einem iterativen Entwicklungsprozess unterzogen, um über einen längeren Entwicklungszeitraum einen bestmöglichen Prototypen zu entwickeln.

Der eigene Beitrag dieses Kapitels besteht in der Umsetzung der Konzeption aus Kapitel 6, die bislang noch nicht existierte. Mit der Implementierung konnte die Machbarkeit der Konzeption nachgewiesen und die Basis für Evaluationen geschaffen werden.

In den nachfolgenden Kapiteln 8 und 9 erfolgt die Evaluation des Annotationssystems und der Methode, um die Effekte von OpenProposal auf eine Anwenderbeteiligung zu evaluieren.

8 Evaluation des OpenProposal Annotationssystems

Ziel dieses Kapitels ist die Evaluierung des OpenProposal Annotationssystems mithilfe von Usability Tests, um die Gebrauchstauglichkeit des Annotationssystems zu erfassen, Verbesserungsmöglichkeiten zu identifizieren und seine Eignung für den Einsatz in Fallstudien sicherzustellen.

Während der Implementierung des Annotationssystems von OpenProposal wurden zu wichtigen Zeitpunkten Usability Tests, u.a. vor einem neuen Release, kurz vor Beginn einer Fallstudie und während den Fallstudien, durchgeführt. Desweiteren wurde die Benutzbarkeit von internen Usability Testern bzw. Evaluatoren aus dem internen Entwicklungsteam regelmäßig auf den Prüfstand gestellt, wie es unter dem Begriff „Heuristic Evaluation“ nach Nielsen [1994b, 155ff] empfohlen wird.

Der erste Benutzertest erfolgte im Februar 2007 mit 15 Anwendern, der zweite im September 2007 im Rahmen der TRUMPF-Fallstudie (vgl. Kapitel 9.4) mit elf Anwendern. Ein dritter Test fand im Juli 2008 während der WAVES-Fallstudie (vgl. Kapitel 9.5) mit 16 Anwendern statt. Ein vierter Test wurde in kleinem Rahmen im September 2008 mit drei wissenschaftlichen Mitarbeitern der Universität Karlsruhe absolviert. Insgesamt konnten somit von 44 unterschiedlichen Anwendern Feedback zu OpenProposal gewonnen und in der Entwicklung berücksichtigt werden. Die vier Usability Tests sind identisch im Untersuchungsgegenstand, in der Evaluationshypothese und im Ablaufschema. Sie unterscheiden sich allerdings in der verwendeten OpenProposal-Version, im Umfang und Auswahl der Aufgaben und in der Auswahl der Teilnehmer.

Im Folgenden werden die Grundlagen zur Vorgehensweise bei Usability Tests beschrieben, die Planung und der Verlauf der Usability Tests von OpenProposal beschrieben und abschließend die Ergebnisse der Tests zusammengefasst sowie Schlussfolgerungen auf die Gebrauchstauglichkeit des Annotationssystems getroffen.

8.1 Zur Methode des Usability Test

Der Usability-Test ist eine der bekanntesten Methoden zur Evaluation der Gebrauchstauglichkeit (vgl. [Sarodnick und Brau, 2006]) und wird oft auch synonym als Benutzbarkeitstest oder Nutzertest bezeichnet. Dabei wird ein System anhand vorgegebener Aufgaben von Benutzern ausprobiert. Sie werden dabei von Usability-Experten beobachtet und während des Tests bei Bedarf befragt. Falls notwendig oder möglich werden ergänzend automatisierte Messungen der Benutzerinteraktion (z.B. Dauer der Benutzung, Häufigkeit von Fehlern, etc.) durchgeführt. Aus den Beobachtungen der Usability-Experten, den Äußerungen der Benutzer und den erfassten Messdaten werden Probleme und Verbesserungsmöglichkeiten abgeleitet. Im Anschluss an einen Usability Test finden im Regelfall eine Befragung der Teilnehmer und eine Diskussionsrunde mit allen teilnehmenden Testern und Experten statt, um die Erfahrungen und die Ergebnisse des Tests in großer Runde auszutauschen.

8.1.1 Usability Tests als Forschungsansatz

Mit einem Usability Tests können nach Rubin et al. [2008] unterschiedliche Ziele erreicht werden:

- Erstellung eines historischen Vergleichswertes für nachfolgende Systemversionen,
- Minimierung der Kosten für Hilfeleistungen,
- Steigerung der Verkäufe und die Chancenerhöhung für wiederholte Verkäufe durch höhere Kundenzufriedenheit,
- Wettbewerbsvorteile durch Abgrenzung von weniger benutzerfreundlichen Produkten,
- Minimierung von Risiken bei der Benutzung von Produkten.

Neben dieser eher praxisnahen Sichtweise stellen Usability Tests auch eine wissenschaftlich relevante Evaluationsmethode dar (vgl. [Sarodnick und Brau, 2006]). In Form einer Evaluation kann ein Usability Test zu einer „*systematischen und möglichst objektiven Bewertung eines geplanten, laufenden oder abgeschlossenen Projektes*“ dienen - mit dem Ziel „*spezifische Fragestellungen zu beantworten und/oder den Zielerreichungsgrad eines bestimmten Vorhabens zu erheben*“ [Sarodnick und Brau, 2006, S. 19]. Für wissenschaftliche Fragestellungen ist der Usability Test aufgrund mehrerer Gründe geeignet, wenn die teilnehmenden Personen sorgfältig ausgewählt und in ausreichender Anzahl (mehr als acht Personen) verfügbar sind:

- Die Vorhersagekraft bzw. die externe Validität eines Usability Tests ist sehr hoch. Die Untersuchungen finden im direkten Dialog mit den betroffenen Zielgruppen statt. Bei den von den Teilnehmern geäußerten Meinungen ist davon auszugehen, dass sie für die gesamte Zielgruppe von großer Bedeutung sind.
- Die Ergebnisse von Usability Tests sind in großen Teilen unabhängig von den Fähigkeiten der Teilnehmer, wenn diese aus unterschiedlichen Gruppen mit unterschiedlichen Fähigkeiten gewählt werden und die Vorgehensweise sorgfältig strukturiert wird.
- Die Objektivität eines Usability Tests ist bei einer sorgfältig geplanten Evaluation hoch. Der Testleiter eines Usability Tests nimmt allerdings im Rahmen eines Usability Tests stets im geringen Maß Einfluss auf die Teilnehmer – auch wenn keine Beeinflussung der Teilnehmer beabsichtigt wird. Im Falle einer wissenschaftlichen Untersuchung sollte daher die Vorgehensweise im Usability Test stark formalisiert werden, um den Einfluss des Testleiters zu begrenzen.
- Die Reliabilität eines Usability Tests ist als hoch zu bewerten. So kann davon ausgegangen werden, dass eine Wiederholung eines Usability Tests ähnliche Ergebnisse hervorbringen wird.

Usability Tests können als induktive oder deduktive Tests durchgeführt werden. In induktiven Tests werden Prototypen bzw. frühe Versionen getestet und dadurch Schwachstellen identifiziert und Verbesserungsmöglichkeiten erarbeitet. Deduktive Tests vergleichen mehrere Alternativen miteinander, beurteilen die Leistungsfähigkeit eines Systems im Vergleich zu etablierten Systemen oder kontrollieren, ob Verbesserungseffekte nach der Implementierung von Verbesserungsvorschlägen eintreten.

Qualitätsmerkmale	Beschreibung
Aufgabenangemessenheit	geeignete Funktionalität, Minimierung unnötiger Interaktionen
Selbstbeschreibungsfähigkeit	Verständlichkeit durch Hilfen / Rückmeldungen
Steuerbarkeit (Dialog)	Steuerung des Dialogs durch den Benutzer
Erwartungskonformität	Konsistenz, Anpassung an das Benutzermodell
Fehlertoleranz	erkannte Fehler verhindern nicht das Benutzerziel, unerkannte Fehler: leichte Korrektur
Individualisierbarkeit	Anpassbarkeit an Benutzer und Arbeitskontext
Lernförderlichkeit	Anleitung des Benutzers, Erlernzeit minimal, Metaphern

Tabelle 8.1: Richtlinien der EN ISO 9241-110 zur Dialoggestaltung von Software

Allerdings sind Usability Tests kein Garant für Benutzerfreundlichkeit oder Produkterfolg. Rubin et al. [2008] räumen ein, dass Usability Tests stets eine künstliche Testsituation darstellen, die sich von der tatsächlichen Nutzungssituation unterscheiden kann. Außerdem repräsentiert die Teilnehmergruppe selten vollständig die gesamte Anwender- bzw. Kundengruppe. Daher ist bei der Durchführung von Usability Tests sorgfältig und präzise vorzugehen, um ein möglichst gutes Ergebnis zu erreichen und die Bedürfnisse der Anwender so gut wie möglich abzudecken.

Das zentrale Gütekriterium einer Software aus Sicht des Anwenders stellt ihre Gebrauchstauglichkeit (vgl. Kapitel 2.6) mit den drei Leitkriterien Effektivität, Effizienz und Zufriedenheit dar, wie sie in der Norm DIN EN ISO 9241 (vgl. [DIN, 2004]) definiert wird. Die Abschnitte DIN EN ISO 9241-11 (Anforderungen an die Gebrauchstauglichkeit) und EN ISO 9241-110 (Grundsätze der Dialoggestaltung) führen zu einer Auflistung etablierter Qualitätsmerkmale (vgl. Tabelle 8.1). Eine Software gilt als den Aufgaben angemessen, wenn sie den Anwender bei der Durchführung seiner Aufgaben geeignet unterstützt und dabei auf unnötige Interaktionen verzichtet. Ferner spielen Kriterien wie die Selbstbeschreibungsfähigkeit, Steuerbarkeit, Erwartungskonformität, Fehlertoleranz, Individualisierbarkeit und Lernförderlichkeit eine entscheidende Rolle für die Gebrauchstauglichkeit.

8.1.2 Vorgehensweise in Usability Tests

Eine anwendungsnahe und ausführliche Anleitung zur Durchführung von Usability Tests findet sich in Rubin et al. [2008] bzw. Rubin [1994]. Weitere ähnliche Vorgehensschema beschreiben Nielsen [Nielsen, 1994a] und Sarodnick und Brau [2006]. Das Ablaufschema eines Usability Tests veranschaulicht Abbildung 8.1.

Bereits die Testumgebung spielt eine wichtige Rolle bei einem Usability Test. Bei der Vorbereitung sollte darauf geachtet werden, dass die Testperson sich trotz einer permanenten Beobachtung wohl fühlen und die zu testende Software ohne Beeinflussung ausprobieren kann. Die Ausstattung einer Umgebung kann von einer einfachen improvisierten mobilen Einrichtung bis hin zu einem professionellen Teststudio variieren. Allerdings ist im Regelfall keine aufwändige Einrichtung notwendig, um

einen Usability Test erfolgreich durchführen zu können. Vielmehr sind nach Rubin et al. [2008] folgende Rahmenbedingungen erfolgsentscheidend:

- Der Test sollte in einem ruhigen und geschlossenen Raum stattfinden. An der Tür sollte ein Hinweis angebracht werden, dass nicht gestört werden darf. Idealerweise ist der Raum bis auf die Ausrüstung für den Test und das übliche Mobiliar frei von jeglicher Ablenkung.

- Im Raum sind nur die für den Test notwendigen Personen anwesend. Für den Test notwendig sind die Testperson, der Testleiter und eine kleine Menge an Beobachtern. Die Testperson sitzt am Computer bzw. dem Testsystem. Der Testleiter sitzt neben ihr. Die Beobachter sitzen in einem großen Abstand zur Testperson und verhalten sich ruhig. Eine oder mehrere Kameras sind optional auf die Testperson und das Testsystem gerichtet. Ein Mikrophon ist am Tisch des Testsystems montiert, um die Kommentare der Testperson aufzunehmen.

- Alternativ können die Beobachter in einem separaten Raum untergebracht werden, falls die Testperson bzw. der Testraum mithilfe mehrerer Videokameras vollständig aus der Ferne beobachtet werden kann. Vorteil ist, dass sich die Testperson nicht das Gefühl hat unter Beobachtung zu stehen.

- Alternativ kann der Testleiter an einem separaten Tisch entweder im gleichen Raum oder in einem separaten Raum sitzen und den Test von der Ferne aus begleiten. Dies hätte den Vorteil, dass die Testperson sich frei bewegen kann. Allerdings ist es möglich, dass sich die Testperson isoliert fühlt und der Testleiter für Fragen schwieriger zu erreichen ist.

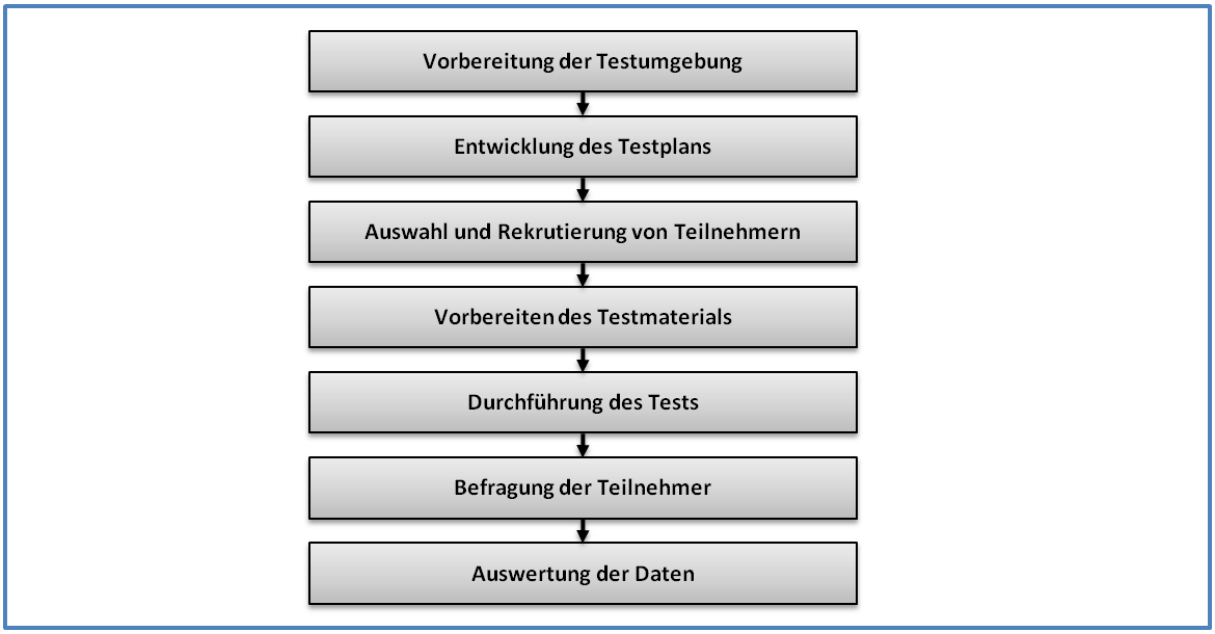

Abbildung 8.1: Ablauf eines Usability Tests [Rubin et al., 2008]

Der Testplan bildet die Grundlage des gesamten Usability Tests. Darin wird das „wie“, „wann“, „wo“, „wer“, „warum“ und „was“ des Testes beschrieben. Er dient als Bauplan für den Ablauf des Tests und hilft bei der Kommunikation zwischen allen am Test beteiligten Personen. Aus dem Plan lässt sich zudem folgern, welche Ressourcen für den Test notwendig werden. Ein Testplan besteht aus mehreren Abschnitten:

- Zweckbestimmung: Sie beschreibt das Anliegen bzw. den Auslöser, warum der Test durchgeführt werden soll.

- Problemstellung: Sie konkretisiert die Zweckbestimmung und beschreibt eine Menge konkreter Fragestellungen, die mit dem Test beantwortet werden sollen. Sie sollte so präzise, exakt, klar und messbar bzw. beobachtbar wie möglich sein.

- Benutzerprofil: Es beschreibt den zukünftigen Anwender bzw. Zielgruppe des Testsystems. Es enthält im Regelfall eine Charakterisierung der Anwender anhand der Eigenschaften „allgemeine Erfahrung mit Computern“, „Bildungsstand“, „Alter“, „Geschlecht“, „Lernsystem“, „Erfahrungen mit unterschiedlichen Betriebssystemen“, etc.

- Methode: Sie beschreibt detailliiert wie der Test mit den Testpersonen durchgeführt werden soll. Darin enthalten sind alle Schritte von der Ankunft der Testpersonen bis zu ihrer Abreise. Der Detailliierungsgrad soll so gewählt werden, dass der Ablauf vollständig nachvollziehbar ist und auch andere Personen diesen Test mithilfe der Beschreibung durchführen könnten.

- Aufgabenliste: Sie beschreibt die Aufgaben, die die Testpersonen während des Tests bearbeiten sollen. Anhand der Aufgaben benutzen die Testpersonen das Testsystem, sammeln ihre Erfahrungen und können schließlich beurteilen, wie gut sie ihre Aufgaben mit dem Testsystem bearbeiten konnten.

- Testumgebung: Sie beschreibt die Umgebung der zukünftigen Nutzung des Systems, die mit der Testumgebung für den Test simuliert werden soll. Darin ist auch das notwendige Material beschrieben, das die Testpersonen üblicherweise benutzen und auch für den Test des Testsystems benötigen.

- Rolle des Testleiters: Sie beschreibt die Aufgaben und Verantwortlichkeiten des Testleiters. Hier können auch Ausnahmesituationen beschrieben und die darauf folgenden Reaktionen des Testleiters definiert werden.

- Evaluationskriterien: Sie beschreiben, welche Arten von Messungen durchgeführt und welche Daten während des Tests erhoben werden. Diese lassen sich in Leistungsdaten („performance measures“) und Präferenzdaten („subjective measures“) unterteilen. Leistungsdaten messen das Verhalten der Testperson, wie z.B. Fehlerraten, benötigte Zeit für eine Aufgabe, etc. Präferenzdaten hingegen beschreiben die Meinung der Testperson, wie z.B. Prioritäten, Antworten auf Fragen, Kommentare, etc. Beide Arten von Daten können quantitativer oder qualitativer Natur sein.

- Inhalte und Form des Ergebnisberichts: Sie beschreiben, wie der Ergebnisbericht des Tests verfasst und an die Entscheider und Entwickler kommuniziert werden soll.

Bei der Auswahl und Rekrutierung der Testpersonen ist darauf zu achten, dass die Testpersonen die zukünftige Zielgruppe so gut wie möglich repräsentieren. Daher ist vorab zu identifizieren, welche Fähigkeiten und Wissen die Zielgruppe besitzt und diese in Form von Benutzerprofilen zu charakterisieren. Anschließend wird die Anzahl der notwendigen Testpersonen bestimmt. Sie hängt u.a. davon ab, welcher Anspruch an die Ergebnisse gestellt wird, welche Ressourcen verfügbar sind, welche Heterogenität die Zielgruppe besitzt und wie viele Testpersonen verfügbar sind. Außerdem sind

Vorbereitungszeit und Dauer des Tests ein entscheidender Faktor. Nach Einschätzung von Nielsen et al. [1994a] sowie Virzi [1992] können mit ca. fünf Testpersonen bereits 80% der Usability Probleme entdeckt werden. Nach den Erfahrungen von Rubin et al. [2008] reichen acht Personen aus. Außerdem sollte mindestens eine „minimal-kompetente“ Testperson („least competent user“), deren Fähigkeiten und Wissen nur die Minimal-Voraussetzung des definierten Benutzerprofils erfüllt, in den Teilnehmerkreis aufgenommen werden. Falls sich diese Person mit dem Testsystem zurechtfindet, ist davon auszugehen, dass auch alle anderen Personen mit dem Testsystem umgehen können. Für statistische Untersuchungen ist eine noch höhere Anzahl von Testpersonen zu empfehlen, um statistisch signifikante Ergebnisse zu erhalten.

Einen weiteren Schwerpunkt der Vorbereitung stellt die Erstellung des Testmaterials dar. Es dient der Kommunikation mit den Testpersonen, der Erhebung von Daten und der Sicherstellung der rechtlichen Rahmenbedingungen. Hierzu gehören u.a. Fragebögen, Teilnehmerinformation, Einverständniserklärung bzw. Geheimhaltungserklärung, Aufgabenbeschreibung, Beobachtungsprotokolle.

Die Durchführung eines Tests fordert neben einer guten Vorbereitung auch eine adäquate Verhaltensweise des Testleiters. So sollte sich der Testleister neutral und zurückhaltend verhalten und für eine angenehme Testatmosphäre sorgen. Die Testpersonen sollten nicht unter Druck gesetzt werden. Auch bei großen Bedienproblemen mit dem Testsystem sollte der Testleiter ruhig bleiben und nicht Partei für das Testsystem ergreifen.

Desweiteren ist bei der Planung der Durchführung zu beachten, dass ein Test unterschiedlich gestaltet werden kann:

- Die Anzahl der Personen, die gleichzeitig an einem Test teilnehmen, kann variieren. Üblicherweise nimmt eine Person pro Test teil. Dies hat den Vorteil, dass sich der Testleiter vollständig auf die eine Testperson konzentrieren kann. Wenn mehrere Personen gleichzeitig teilnehmen, kann zwar Zeit gespart aber dafür können auch weniger Beobachtungen notiert werden.

- Die Testpersonen können in Gruppen eingeteilt werden, falls unterschiedliche Systeme bzw. Komponenten getestet werden sollen.

- In einem frühen Stadium der Software-Entwicklung ist die Interaktion zwischen Testperson und Testleiter sehr intensiv, da zunächst nicht Leistungsdaten sondern Präferenzdaten erfasst werden und die Effektivität des Systems untersucht wird. In späteren Tests nimmt der Intensitätsgrad der Interaktion ab, wenn in den Tests vor allem Leistungsdaten erhoben werden und die Effizienz eines Systems im Mittelpunkt steht.

- Daten können auf mehreren Wegen erhoben werden. Videokameras können zur Aufzeichnung des Verhaltens der Testperson eingesetzt werden. Eine Aufnahmesoftware dient zur Aufzeichnung des Computerbildschirms. In der Testsoftware können alle Benutzerinteraktionen protokolliert werden. Der Testleiter und die Beobachter können ein Beobachtungsprotokoll anfertigen. Zudem kann der Testleiter bei Bedarf Fragen an die Testperson stellen. Außerdem hat sich die Methode des „Laut Denkens“ („thinking aloud“) als hilfreich erwiesen, bei der die Testperson ihre Gedanken laut ausspricht und somit Eindrücke, Denkweise, etc. preisgibt. Allerdings kann das „Laut Denken“ zu einer Belastung der Testperson führen, so

dass der Testperson die Bearbeitung der Aufgaben erschwert wird. Dies verträgt sich somit nicht mit der Erfassung von Leistungs- bzw. Performanz-Daten. Zudem kann sich die Testperson dabei unwohl fühlen.

Nach der Durchführung des Tests ist eine abschließende Befragung der Testperson erfolgsentscheidend, um die Probleme vollständig zu verstehen und Lösungsmöglichkeiten zu diskutieren. Hierzu können unterschiedliche Techniken angewendet werden. Zu Beginn sollte der Testleiter die Testperson ihre Gedanken aussprechen lassen. Anschließend kann er Fragen zum Ablauf des Tests stellen und bei den aufgetretenen Problemen nachhaken. Wenn möglich können auch ausgewählte Videosequenzen des Tests abgespielt werden, um die Testperson bei der Erinnerung an die Situation zu unterstützen. So kann man bei besonderen Problemen besser über die Hintergründe der Probleme sprechen. Eine andere Technik sieht vor, dass die Testperson nach derjenigen Situation zu fragen, an die sie sich am besten erinnern kann. Mit dieser Technik kann man prüfen, ob sich Elemente des Testsystems als hilfreich oder störend erweisen oder evtl. nicht wahrgenommen werden. Für besonders wichtige Elemente eines Systems kann der Testleiter seine neutrale Haltung verlassen und die Gegenposition zur Testperson einnehmen – ohne jedoch aggressiv auf die Testperson zu wirken. Mithilfe dieser Technik wird die Testperson motiviert, ihre Position zu überlegen und zu verteidigen.

Nach Abschluss des Tests erfolgt die Auswertung der Daten, die Ableitung von Verbesserungsvorschlägen und die Erstellung des Ergebnisberichts. Grundlage der Auswertungen stellen die anfangs beschriebene Problemstellung und die Evaluationskriterien zum Usability Test dar. Sie werden bei der Sichtung des Datenmaterials herangezogen und erleichtern die Strukturierung des Datenmaterials.

Zur Erleichterung der Analyse können alle Sprach- und Videoaufnahmen transkribiert werden. Alternativ können zu den Aufnahmen auch Zusammenfassungen erstellt werden, in denen alle Daten zu den Evaluationskriterien enthalten sind. Außerdem sollten auch die Notizen aus den Beobachtungsprotokollen und die Antworten auf den Fragebögen transkribiert werden, um alle Daten elektronisch weiterverarbeiten zu können. Anschließend können Fehlerquellen zu den Daten identifiziert und mögliche Lösungsansätze diskutiert werden. Zudem sollte auch offen gelegt werden, welche Probleme nicht eindeutig bestimmbar und für die somit weitere Untersuchungen notwendig sind.

8.2 Planung der Usability Tests

Untersuchungsgegenstand der Usability Tests bildete der Grad der Benutzerakzeptanz zur Bedienoberfläche, zum Bedienkonzept und zu den Zielen von OpenProposal. Zentrale Fragestellung der Tests war, ob OpenProposal von den Benutzern akzeptiert wird, und wie die Gestaltung von OpenProposal weiter verbessert und an die Arbeitsweise und Anforderungen der Benutzer angepasst werden kann. Die Gebrauchstauglichkeit von OpenProposal drückt sich dadurch aus, dass mit OpenProposal Verbesserungsvorschläge auf eine effiziente und effektive Art und Weise erstellt werden können und dass die Bedienung von den Benutzern angenehm bzw. positiv wahrgenommen wird.

8.2.1 Ablauf der Usability Tests

Die Gestaltung der Usability Tests orientierte sich an der von Nielsen [1994a] bzw. Sarodnik und Brau [2006] und Rubin et al. [2008] empfohlenen Vorgehensweise, wie sie in Kapitel 8.1 beschrieben wird.

Da OpenProposal für die Erstellung von Verbesserungsvorschlägen konzipiert wurde, konnte es nicht für sich alleine getestet werden. Es war daher notwendig, dass OpenProposal für die Evaluation bei einem Usability Test einer beliebigen Software eingesetzt wird.

Jeder Usability Test für OpenProposal folgte dabei demselben Ablaufschema:

1. Der Benutzer wurde um eine freiwillige Teilnahme gebeten und über den Ablauf des Benutzertests informiert. Bei der Auswahl der Benutzer wurde sichergestellt, dass der Benutzer nicht an der Entwicklung von OpenProposal beteiligt und mit OpenProposal vor dem Test nicht vertraut war.

2. Der Benutzer saß an einem Computer, auf dem OpenProposal und eine zu testende Software zum Benutzen bereitstehen. Der Testleiter war im gleichen Raum anwesend und bemühte sich um eine geringmögliche Beeinflussung des Benutzers. Die Bedienung des Computers erfolgte während des Tests ausschließlich durch die Testperson.

3. Zu Beginn wurde dem Benutzer OpenProposal kurz vorgeführt und die Funktionsweise der Annotationswerkzeuge anhand von Beispielen demonstriert.

4. Anschließend erhielt die Testperson ein Blatt mit mehreren Aufgaben, die mit der zu testenden Software in einer vorgegebenen Zeit zu lösen waren. Er wurde gebeten, auf Verbesserungsmöglichkeiten der getesteten Software zu achten und diese mit OpenProposal als Anwenderbeitrag festzuhalten.

5. Während des Tests wurde das Benutzerverhalten beobachtet und protokolliert. Der Benutzer hatte jederzeit die Möglichkeit, den Testleiter um Rat zu fragen.

6. Nach Abschluss des Tests wurde dem Benutzer ein Fragebogen ausgeteilt. Im Fragebogen wurde um statistisch-relevante und demografische Angaben zur Person und um die Bewertung von OpenProposal gebeten. Der Fragebogen wurde in Abwesenheit des Testleiters ausgefüllt. Die Gestaltung der Fragebogen orientierte sich an den Richtlinien und Gestaltungsregeln von Schnell et al. [2008].

7. Zur Auswertung der Tests wurden die erstellen Verbesserungsvorschläge, die Beobachtungsprotokolle sowie die Fragebögen herangezogen.

8.2.2 Struktur des Fragebogens der Usability Tests

Ziel des Fragebogens war es, die Testpersonen zu Verbesserungsmöglichkeiten von OpenProposal und zur Effizienz und Effektivität der Erstellung von Verbesserungsvorschlägen zu befragen. Es wurde vorausgesetzt, dass die Testperson kurz zuvor mit OpenProposal gearbeitet und OpenProposal zur Erstellung von Verbesserungsvorschlägen verwendet hat.

Der Fragebogen gliederte sich in vier Teile. Im ersten Teil wurde der Tester in einem Textblock über die Ziele und Aufbau des Fragebogens informiert und die Anonymisierung zugesichert. Außerdem wurden wichtige Hinweise zum Ausfüllen des Fragebogens gegeben. Im zweiten Teil wurden einleitend allgemeine Angaben zur Person erfragt. Hierzu gehörten Fragen zur Berufserfahrung, zum Geschlecht, zur Tätigkeit bzw. Fachrichtung und zu testrelevanten Handicaps. Ferner wurde die Testperson nach ihren bisherigen Erfahrungen zur Formulierung von Anforderungen befragt und um eine

Selbsteinschätzung ihrer Fähigkeit zur Formulierung von Anforderungen gebeten. Außerdem wurden Fragen zur Vertrautheit mit der getesteten Software gestellt.

Der dritte Teil bezog sich auf die Gestaltung der grafischen Benutzerschnittstelle und fragte nach den Erfahrungen mit der Bedienung und Handhabung von OpenProposal. Neben der Möglichkeit allgemeine Kommentare abzugeben, konnte die eigene Meinung zu vorformulierten Thesen auf einer Likert-7-Punkte-Skala abgegeben werden. Als Thesen wurden folgende Aussagen zur Diskussion gestellt:

- „Ich habe mich schnell in OpenProposal zurechtgefunden."
- „Ich bin bei OpenProposal oft stecken geblieben und musste einen anderen Weg suchen."
- „OpenProposal zeigt zu viele Informationen auf einmal; das hat mich verwirrt."
- „Vorschläge sind einfach und ohne Nachdenken zu müssen zu erstellen."
- „Ich habe Fehler beim Einsatz von OpenProposal gemacht."
- „Das Erstellen von Vorschlägen war unnötig kompliziert und langwierig."
- „Insgesamt macht die Software einen ausgereiften Eindruck."

Als Antwortmöglichkeiten standen folgende Optionen zur Verfügung: „7 = Trifft voll und ganz zu", „6 = Trifft größtenteils zu", „5 = Trifft eher zu", „4 = Weder noch", „3 = Trifft eher nicht zu", „2 = Trifft größtenteils nicht zu", „1 = Trifft überhaupt nicht zu".

Zum Schluss wurde der Tester um eine Gesamtbenotung von OpenProposal nach dem Schulnotensystem gebeten. Bei Bedarf konnten die Testpersonen auf jeder Seite des Fragebogens in der Fußzeile zusätzlich allgemeine Kommentare zum Test und zum Fragebogen abgeben.

8.3 Ergebnisse der Usability Tests

Die Ergebnisse der Usability Tests ergaben sich aus der Auswertung der Fragebögen, Beobachtungsprotokolle und Videoaufzeichnungen. Insgesamt konnten 36 Tests mit vollständig ausgefüllten Fragebögen und mit Beobachtungsprotokollen und Videoaufzeichnungen über jeweils ca. 90 Minuten durchgeführt werden. Bei acht Teilnehmern wurden die Fragebögen nicht vollständig ausgefüllt, so dass deren Tests und Daten nicht in die Auswertung einflossen.

8.3.1 Ablauf der Usability Tests am FZI

Der erste Usability Test im Februar 2007 fand mit 15 Mitarbeitern und studentischen Hilfskräften des FZI am FZI Forschungszentrum Informatik statt. OpenProposal stand in Version 1.0 zur Verfügung. Jeder Test erfolgte in Form eines Einzelinterviews und war auf 90 Minuten terminiert. Nach einer kurzen Einführung wurden die Testpersonen gebeten, fünf Aufgaben mit der Anwendung Mozilla Firefox zu lösen und ihre Verbesserungsvorschläge zu Firefox mit OpenProposal zu skizzieren. Dabei sollen die Tester die „thinking aloud"-Methode anwenden und somit ihre Wahrnehmungen und Eindrücke während des Tests aussprechen. Die Tests wurden mit einer Videokamera und einer Bild-

schirmaufnahmesoftware aufgezeichnet. Außerdem erstellte der Testleiter handschriftliche Notizen. Am Ende des Tests wurden die Testpersonen gebeten, einen Fragebogen auszufüllen.

Die Testaufgaben bezogen sich auf die Software Mozilla Firefox in der alten Version 1.0. Die Probanden wurden aufgefordert, mit OpenProposal Anwenderbeiträge für den Firefox-Browser zu formulieren. Die ersten drei Aufgaben bezogen sich auf konkrete Anforderungen, die in Firefox in Version 2.0 umgesetzt wurden. Damit sollten Anforderungen nachgebildet werden, die es tatsächlich gab. Die letzten beiden Aufgaben ließen den Probanden viel Freiraum und forderten sie auf, selbstständig Vorschläge zu machen. Damit sollen zum einen durch die in den Aufgaben 1-3 konkret vorgegeben Vorschlagsinhalte vergleichbare Ergebnisse erzielt und zum anderen durch die in den Aufgaben 4 und 5 relativ freie Fragestellung die offene Nutzung von OpenProposal untersucht werden. In Aufgabe 1 sollte der Proband einen Menüpunkt „Automatic Update“ einfügen. Aufgabe 2 ließ den Probanden Vorschläge zum „Tabbed-Browsing“ (Drag&Drop, Knopf „Tab Schließen“ an jedem Tab-Register anzeigen und konfigurieren) formulieren. Für Aufgabe 3 sollte der Proband das Menü „Tools“ („Read Mail“ und „New Message“ streichen, „Extensions“ und „Themes“ zusammenfassen, „Javascript-Console“ umbennenen) und das Fenster „Options“ (Neue Bereiche definieren, Bereich „Privacy“ ins Menü „Tools“ verschieben) verbessern. Die letzten beiden Aufgaben gaben dem Probanden einen größeren Spielraum. In Aufgabe 4 wurde der Proband aufgefordert, den RSS-Feed von Spiegel-Online zu abonnieren. Sollten dem Proband Schwierigkeiten beim Einrichten des RSS-Abo auffallen, wurde er gebeten einen Anwenderbeitrag mit dem OpenProposal Annotationssystem zu formulieren. In Aufgabe 5 sollte der Proband gebeten, drei weitere eigene Änderungsvorschläge zu Firefox zu formulieren. Zum Schluss wurden die Anwender gebeten, einen Fragebogen zu OpenProposal auszufüllen.

8.3.2 Ablauf der Usability Tests bei TRUMPF

Der zweite Usability Test im September 2007 erfolgte mit elf Testpersonen bei dem Unternehmen TRUMPF im Rahmen des Usability Workshops der Fallstudie TRUMPF (vgl. Kapitel 9.4). Die Testpersonen waren Mitarbeiter von TRUMPF und stammten aus unterschiedlichen Abteilungen. Ziel des Workshops war es eine neue Software der TRUMPF zu testen und die Eignung von OpenProposal für die Erfassung von Verbesserungsvorschlägen der Anwender zu evaluieren. Neben den Untersuchungen in der Fallstudie wurde auch die Gebrauchstauglichkeit von OpenProposal gemäß der Vorgehensweise eines Usability Tests untersucht.

Der Workshop wurde über zwei Arbeitstage mit allen Testpersonen in einem großen Workshop-Raum durchgeführt. Jeder Testperson wurde ein eigener Computer zugewiesen, auf dem OpenProposal in Version 2.1 und die zu testende Software aufgespielt war. Nach einer kurzen Einführung in den Ablauf des Workshops und einer Demonstration der Funktionsweise von OpenProposal wurde mit dem Workshop begonnen. Zwei Forscher notierten im Hintergrund ihre Beobachtungen zur Gebrauchstauglichkeit und standen den Benutzern für Fragen, Anmerkungen und technischen Problemen zur Verfügung. Am Ende jedes Workshoptages gab es eine gemeinsame Diskussion über die Eindrücke des vergangenen Tages und die Testpersonen wurden gebeten, einen Fragebogen auszufüllen.

Bei der getesteten Software handelte es sich um ein komplexes 3D-Planungsintrument zur Konstruktion von 3D-Modellen von Maschinenteilen. Als Aufgaben wurden Konstruktionszeichnungen vorge-

legt, die den typischen Tätigkeiten ihrer Kunden entsprechen. Mit OpenProposal sollten die Testpersonen ihre Verbesserungsvorschläge an der 3D-Planungssoftware von TRUMPF dokumentieren.

8.3.3 Ablauf der Usability Tests bei WAVES

Der dritte Usability Test fand im Juli 2008 als Teil der Fallstudie WAVES statt (vgl. Kapitel 9.5, Seite 188ff) mit Mitarbeitern der Anwendungspartner des Forschungsprojekts WAVES. In den ersten zwei Wochen der Fallstudie führte der Moderator Einzelinterviews mit 16 Anwendern bei den Unternehmen durch. Während der Einzelinterviews testete jeder Anwender zwei Prototypen des Forschungsprojektes an einem Computer, wobei er gebeten wurde, die „thinking aloud" Methode anzuwenden und Verbesserungsvorschläge zu den Prototypen mit OpenProposal in der Version 2.43 zu verfassen.

Bei den beiden Prototypen handelte es sich um die Anwendungen Wiquila und WavesIS. Wiquila ist ein Wiki-Editor, welcher u.a. an mehrere Wikis angebunden werden kann, über eine WYSISWG-Ansicht verfügt, und eine Autovervollständigen-Funktion anbietet. WavesIS ist eine Suchmaschine, mit der gleichzeitig in verschiedenen Informationsquellen wie z.B. Wikis, Versionsverwaltungssystemen, etc. gesucht werden können. Beide Prototypen wurden in den Unternehmen installiert und in deren Systeme integriert. Der Moderator saß neben der Testperson und gab gering strukturierte Aufgaben vor. In den Testaufgaben mussten die Anwender typische Tätigkeiten zum Suchen und Bereitstellen von Informationen mit den Prototypen verrichten. Die Interviews wurden mit einer Videokamera und einer Bildschirmaufnahmesoftware aufgezeichnet. Nach dem Interview wurde jeder Anwender gebeten, einen Fragebogen zu OpenProposal auszufüllen.

8.3.4 Ablauf der Usability Tests an der Universität Karlsruhe

Ein vierter Test wurde im September 2008 am Karlsruher Institut für Technologie (ehemalige Universität Karlsruhe) mit drei wissenschaftlichen Institutsmitarbeitern aus der Fakultät für Wirtschaftswissenschaften absolviert. An ihrem Institut wurde in diesem Zeitraum eine neue Internetplattform mit einem Redaktionssystem zur Präsentation des Instituts im Internet eingeführt, die im Rahmen eines Usability Tests mit OpenProposal in der Version 2.43 getestet werden soll. Die Tests waren als Einzelinterviews auf 90 Minuten angesetzt. Während des Tests war ein Testleiter anwesend, der die Anwender in den Test einwies, sie beim Testen beobachtete und ihnen zum Schluss einen Fragebogen austeilte.

8.3.5 Auswertung der Ergebnisse der Usability Tests

Abbildung 8.2 beschreibt die demografische Zusammensetzung der 36 Teilnehmer. 27 Personen waren Männer, neun waren Frauen. Der Altersdurchschnitt lag bei 31,58 Jahre (MW: 31,58, Median: 29), wobei die meisten Teilnehmer zwischen 28 und 32 Jahre alt waren (25% Quartil: 28, 75% Quartil: 32,25). Handicaps besaß keine der Testpersonen. Die Mehrheit der Teilnehmer (33%) war auch im beruflichen Umfeld Anwender. Fast ebenso viele übten den Beruf des Software-Entwicklers (31%) oder Entscheiders (28%) aus. Drei Teilnehmer (8%) waren beruflich als Moderatoren aktiv. 23 Teilnehmer (64%) gaben an, bereits Erfahrungen mit der Formulierung von Anforderungen gesammelt zu haben. Die Mehrheit (42%) schätzen sich selbst als Experte bei der Formulierung von Anforderungen ein. Zwölf Teilnehmer (25%) gaben an Anfänger zu sein, und neun Teilnehmer (25%) schätzen ihre Fähigkeiten als durchschnittlich ein. Die im Usability Test getestete Software wurde von der Mehrheit

täglich (53%) bzw. mehrmals in der Woche (17%) benutzt. Elf Teilnehmer (30%) kannten die Software nicht bzw. benutzten sie selten.

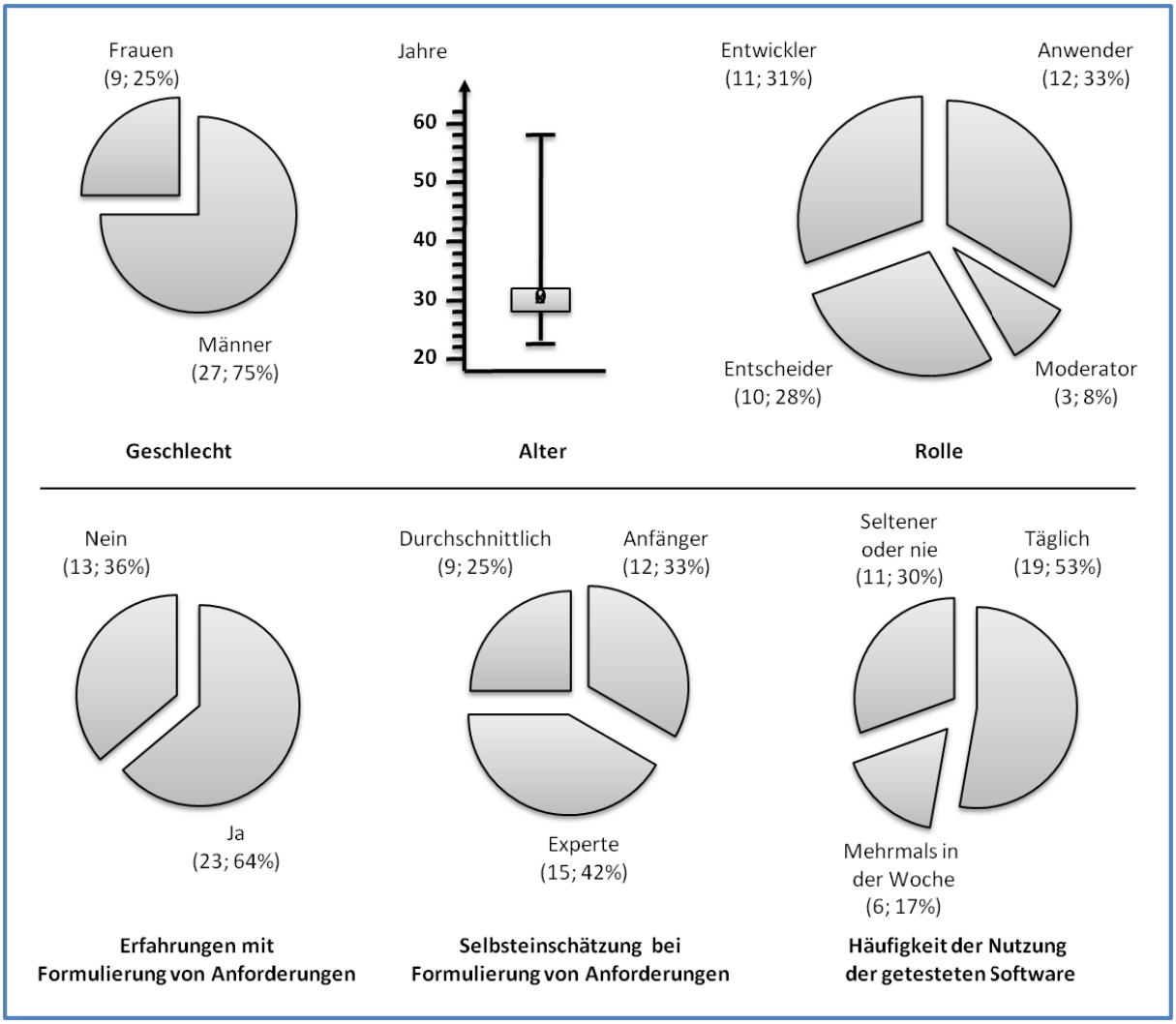

Abbildung 8.2: Demografische Daten der Teilnehmer der Usability Tests

Tabelle 8.2 fasst die Antworten auf die Thesen aus dem Fragebogen zusammen („1 = Trifft überhaupt nicht zu“, „4 = Weder noch“, „7 = Triff voll und ganz zu“, „N“ = Anzahl der Teilnehmer). Demnach fanden sich alle Testpersonen schnell in OpenProposal zurecht. Sie konnten mit OpenProposal Beiträge einfach und ohne Nachdenken erstellen. Die Testpersonen widersprachen den Thesen, dass OpenProposal unübersichtlich und kompliziert bzw. langwierig zu bedienen wäre. Allerdings zeigt die Varianz, dass einige wenige der Testpersonen Probleme bei der Bedienung hatten.

Über die unterschiedlichen Versionen von OpenProposal hinweg finden sich vor allem Unterschiede in den Antworten von der Version 1.0 auf Version 2.10. Es ist dabei eine Tendenz der Verbesserung erkennbar. Anhand der Daten kann dem OpenProposal Annotationssystem eine gute Gebrauchstauglichkeit attestiert werden. Die Anwender haben sich schnell in OpenProposal zurechtfinden können (Minimum: 3,00 Maximum: 7,00 Mittelwert: 6,15, Varianz: 0,72), Vorschläge einfach und ohne Nachdenken erstellen können (Minimum: 3,00 Maximum: 7,00 Mittelwert: 5,83, Varianz: 1,09) und einen ausgereiften Eindruck wahrgenommen (Minimum: 4,00 Maximum: 7,00 Mittelwert: 5,82, Varianz: 0,57). Große Probleme hatten die Anwender demnach nicht.

Thesen	Version	N	MW	VAR	MIN	MAX
Ich habe mich schnell in OpenProposal zurechtgefunden.	Alle	36	6,15	0,72	3,00	7,00
	1.00	16	6,00	1,43	3,00	7,00
	2.10	11	6,25	0,21	6,00	7,00
	2.43	19	6,26	0,27	6,00	7,00
Ich bin bei OpenProposal oft stecken geblieben und musste einen anderen Weg suchen.	Alle	36	2,27	2,20	1,00	6,00
	1.00	16	2,52	2,84	1,00	6,00
	2.10	11	1,38	1,13	1,00	4,00
	2.43	19	2,54	1,70	1,00	5,00
OpenProposal zeigt zu viele Informationen auf einmal; das hat mich verwirrt.	Alle	36	2,03	1,80	1,00	7,00
	1.00	16	2,60	3,26	1,00	7,00
	2.10	11	1,25	0,21	1,00	2,00
	2.43	19	1,84	0,47	1,00	3,00
Vorschläge sind einfach und ohne Nachdenken zu müssen zu erstellen.	Alle	36	5,83	1,09	3,00	7,00
	1.00	16	5,53	1,98	3,00	7,00
	2.10	11	5,75	0,21	5,00	6,00
	2.43	19	6,25	0,38	5,00	7,00
Ich habe Fehler beim Einsatz von OpenProposal gemacht.	Alle	36	2,22	1,09	1,00	5,00
	1.00	16	2,47	0.98	1,00	5,00
	2.10	11	2,13	0,70	1,00	4,00
	2.43	19	2,00	1,50	1,00	5,00
Das Erstellen von Vorschlägen war unnötig kompliziert und langwierig.	Alle	36	1,57	0,50	1,00	3,00
	1.00	16	1,80	0,60	1,00	3,00
	2.10	11	1,13	0,13	1,00	2,00
	2.43	19	1,58	0,49	1,00	3,00
Insgesamt macht die Software einen ausgereiften Eindruck.	Alle	36	5,82	0,57	4,00	7,00
	1.00	16	5,87	0,55	4,00	7,00
	2.10	11	5,57	0,95	4.00	7,00
	2.43	19	5,92	0,45	5,00	7,00

Tabelle 8.2: Antworten zur Gebrauchstauglichkeit aus den Usability Tests von OpenProposal

Bei der Gesamtbewertung von OpenProposal (vgl. Tabelle 8.3) wurde in neun Fällen die Note „1,00“, in 26 Fällen eine „2,00“ und in einem Fall eine „3,00“ vergeben. Daraus leitet sich als Median eine „2“ (Minimum: 1,00; Maximum: 3,00; 25%-Quantil: 1,75; 75%-Quantil: 2,00; Mittelwert: 1,78; Varianz: 0,23) ab. Die Gesamtbewertung verbessert sich von Version zu Version minimal, wobei eine Bewertung zwischen „1,00“ und „2,00“ bereits als sehr positiv bewertet werden kann.

Version	N	Median	25%-Quantil	75%-Quantil	MW	VAR	MIN	MAX
Alle	36	2,00	1,75	2,00	1,78	0,23	1,00	3,00
1.00	16	2,00	1,50	2,00	1,80	0,31	1,00	3,00
2.10	11	2,00	2,00	2,00	1,88	0,13	1,00	2,00
2.43	19	2,00	1,00	2,00	1,69	0,23	1,00	2,00

Tabelle 8.3: Antworten zur Gesamtbewertung aus den Usability Tests von OpenProposal

Aus den abgegebenen Kommentaren auf den Fragebögen und den Beobachtungen konnten zahlreiche Anregungen für die Verbesserung der grafischen Benutzeroberfläche, der Bedienung und der Funktionalität gewonnen werden. Folgende Ergebnisse flossen in die Entwicklung mit ein:

- Die Abgabe eines Vorschlags sollte nach der Meinung der Testpersonen so einfach wie möglich sein. So waren in früheren Versionen von OpenProposal mehrere Schritte notwendig, bis ein Vorschlag abgesendet werden konnte. In der finalen Version erfolgt dies in nur einem Schritt. Hierfür musste in Kauf genommen werden, dass der Anwender weder den automatisch generierten Titel, noch die automatisch generierte allgemeine Beschreibung manuell korrigieren oder seinen Vorschlag manuell einer Kategorie zuordnen kann. Stattdessen wird darauf vertraut, dass die automatische Generierung und die Kontexterfassung ausreichen.

- Es wurde der besseren Übersicht willen auf eine Vielzahl von Zusatzfunktionen verzichtet, die zwar als interessant aber nicht als essentiell bewertet wurden und auch während der Tests überhaupt nicht oder selten genutzt wurden. So enthielten frühere Versionen die Möglichkeit freie Zeichnungen wie z.B. Linien, Freitext, Pfeile, Ellipsen, Kreise, etc. einzufügen. Außerdem konnten vorgegebene Standardobjekte wie z.B. Listen, Tabellen, Schaltflächen, etc. eingefügt werden. Kurzzeitig wurde auch eine Funktion zum Kopieren und Einfügen von Bereichen des Bildschirmfotos implementiert, mit der man vorhandene grafische Elemente auf dem Bildschirm kopieren konnte. Ferner wurde auf eine Funktion zum Anhängen von Dateien an den Vorschlag verzichtet, da nur zwei Anwender diesen Wunsch geäußert hatten und andere Anwender keinen Mehrwert in dieser Funktion sahen.

- Im Hauptmenü wurden zahlreiche Funktionen z.B. „Konfigurationsmodus“, „Neuer Vorschlag“, „Neues Bildschirmfoto“, etc. untergebracht, die in vorherigen Versionen direkt von der Benutzeroberfläche ausgewählt werden konnten. Es hat sich gezeigt, dass die Anwender in erster Linie die Annotationswerkzeuge und die Schaltfläche „Absenden“ nutzen und selten andere Funktionen benötigen.

- Zum Anonymisieren standen dem Anwender in früheren Versionen zwei Möglichkeiten zur Verfügung. Beibehalten wurde die Funktion, mit der man bestimmte Bereiche auf dem Bildschirm mithilfe eines grauen Rechtecks ausblenden kann. Nicht beibehalten wurde hingegen die Möglichkeit, den Anzeigebereich des Bildschirmfotos einstellen zu können.

- In früheren Versionen wurde dem Anwender ein zusätzliches Fenster mit einer Liste aller annotierten Objekte angezeigt. Darüber konnte er alle annotierten Objekte überschauen, verwalten und anpassen, bevor er seinen Vorschlag versendet. Das Fenster fand bei den Anwendern kaum Beachtung und sie zogen es vor, die annotierten Objekte direkt zu manipulieren.

- Es wurde diskutiert, ob neben dem Bildschirmfoto auch Video- und Audioaufzeichnungen ermöglicht werden sollten. In den Diskussionsrunden wurde der Mehrwert dieser Funktion in Frage gestellt. Die Annotation von Bildschirmfotos wurde bereits als ausreichend empfunden.

- Auf Wunsch der Anwender wurde eine Tastatursteuerung implementiert, so dass fast alle Funktionen auch über Tastaturkürzel aktiviert werden können. Dies war vor allem für versierte Benutzer von großer Bedeutung.

Neben den Verbesserungsvorschlägen wurden von den Testpersonen auch zahlreiche positive Aussagen getroffen. Bei den Beobachtungen war bei den Testern der Spaß an der Benutzung zu spüren. Die auf den Fragebogen notierten Kommentare machen dies deutlich:

- *„bessere Visualisierung als nur textliche/sprachliche Umschreibung; man sieht schneller und präziser, was sich der Kunde wünscht“*

- *„intuitiv bedienbar, einfach bedienbar, eindeutige Dokumentation“*

- *„Durch ‚Visualisierung‘ sind Vorschläge besser zu verstehen und leichter umzusetzen ‚Ein Bild sagt mehr als 1000 Worte‘“*

- *„verständlicher, einfacher, gute Kommunikation von Kunden, professionell, motivierend“*

8.4 Fazit zu Kapitel 8

Ziel von Kapitel 8 war die Evaluation des OpenProposal Annotationssystems mithilfe von Usability Tests. Von insgesamt 44 Testpersonen gaben 36 Testpersonen vollständige Fragebögen ab. Die Auswertung der Fragebögen ergab, dass OpenProposal einen positiven Eindruck hinterlassen konnte und die Idee der Annotation von Bildschirmfotos für die Erstellung von Anwenderbeiträgen als sinnvoll bewertet wurde. Es zeigte sich, dass sich die Anwender mit dem Konzept der Annotation von Bildschirmfotos zurechtfanden und erfolgreich Anwenderbeiträge erstellen konnten.

Durch die Usability Tests konnten zudem zahlreiche Hinweise auf die Entwicklung herauszogen werden, bei der zahlreiche Funktionen neu implementiert bzw. angepasst sowie zahlreiche Funktionen auch entfernt wurden. Die Antworten zu den Thesen und der Gesamtbewertung zeigen deutlich, dass OpenProposal die Anforderung der Gebrauchstauglichkeit an Effizienz, Effektivität und Zufriedenheit erfüllen kann. Die Anwender gaben an, dass ihnen die Erstellung von Anwenderbeiträgen leicht fiel, sie keine Fehler machten und sie mit dem Ansatz im Gesamten zufrieden sind.

Der Vergleich der unterschiedlichen Versionen miteinander ist nur eingeschränkt möglich. Trotz der massiven Änderungen an der grafischen Oberfläche, der Robustheit und der Bedienung von OpenProposal, unterscheiden sich die Bewertungen zwischen Version 1.00 und Version 2.43 nur

marginal. Dies ist darauf zurückzuführen, dass die Versionen nicht einem direkten Vergleich unterzogen und nicht von den gleichen Testpersonen getestet wurden, so dass die Verbesserungen bzw. Unterschiede nicht explizit wahrgenommen werden konnten.

Da die Teilnehmer der Usability Tests zum großen Teil aus dem Umfeld von Software-Projekten stammen und das Durchschnittsalter um die 30 Jahre beträgt, sind die Ergebnisse nicht repräsentativ für Anwender aller Altersstufen und Berufsfelder. Es ist zu erwarten, dass vor allem ältere Personen ohne Computerkenntnisse eine Einarbeitungszeit in OpenProposal benötigen.

Mit diesem Kapitel konnte nachgewiesen werden, dass OpenProposal für den Einsatz in realen Software-Projekten geeignet ist und keine Abwehrreaktion bei den Anwendern auslöst. Die Anwender sehen durch den Ansatz eine Arbeitserleichterung und konnten sich schnell und leicht mit dem Bedienkonzept anfreunden. Für die weitere Evaluation ist somit die Untersuchung im Rahmen von Software-Projekten durchzuführen und neben der Belastung der Anwender auch die Qualität der Anwenderbeiträge und die Effizienz der Entwicklung zu betrachten. Im nächsten Kapitel 9 wird daher die gesamte OpenProposal Methode erprobt und der Frage nach der Eignung von OpenProposal als Methode zur asynchronen Anwenderbeteiligung nachgegangen.

9 Evaluation der OpenProposal Methode

Ziel dieses Kapitels ist die Evaluation der OpenProposal Methode. Mit der Evaluation sollen die Effekte der Methode untersucht und die Vor- und Nachteile der Methode analysiert werden. Im Unterschied zur Evaluation des OpenProposal Annotationssystems in Kapitel 8 soll die Evaluation der OpenProposal Methode in einem realen Projektrahmen durchgeführt werden, um die Eignung der Methode für Software-Projekte anwendungsnah zu prüfen. Hierfür werden zunächst die Methode der vergleichenden Fallstudien als geeignete Evaluationsmethode motiviert und beschrieben, anschließend die Fallstudien TRUMPF und WAVES geplant und durchgeführt, und schließlich die Ergebnisse der Fallstudien zusammengefasst.

9.1 Auswahl der Methode zur Evaluation

Für die Evaluation von Methoden und Technologien im Rahmen von realen Software-Projekten existieren nach Kitchenham und Pickard [1995; 1996] u.a. die Evaluationsmethoden Experiment („experiment“), Fallstudie („case study“) und Umfrage („survey“). Eine Umfrage kann zum Zeitpunkt der Erhebung eine breite und große Anzahl von Menschen bzw. Projekten erreichen und somit eine große Menge an Daten über die Meinungen und wahrgenommenen Änderungen bzw. Verbesserungen erfassen. Mit Experimenten können Situationen in einer Laborumgebung nachgestellt werden, um alle Rahmenbedingungen kontrollieren und nach Bedarf ändern zu können. Somit können Änderungen an einer Methode oder Technologie exakt gemessen werden. Fallstudien erheben den Verlauf von Software-Projekten über einen längeren Zeitraum in realen Umgebungen. Eine Fallstudie bedient sich dabei Methoden zur Beobachtung und Befragung von Menschen bzw. Organisationen sowie der Analyse von Daten, die im Rahmen der Projekte entstehen. Allerdings besitzen Fallstudien nicht die wissenschaftliche Generalisierbarkeit und Robustheit von Experimenten und Umfragen.

Um die Auswahl einer Evaluationsmethode treffen zu können, bedarf es einer gründlichen Einschätzung der Vor- und Nachteile der Methoden, der Ziele der Evaluation sowie der gegebenen Voraussetzungen und Möglichkeiten. Diese werden im Folgenden definiert.

Ziel der Evaluation ist die Untersuchung der Eignung der OpenProposal Methode zur asynchronen Anwenderbeteiligung in Software-Projekten anhand der Bewertungskriterien aus Kapitel 2.9. Voraussetzung ist somit, dass die Evaluation im Rahmen einer asynchronen Anwenderbeteiligung in Software-Projekten erfolgt und OpenProposal dabei mit anderen synchronen und asynchronen Methoden verglichen wird. Die Möglichkeiten der Evaluation sind durch die Forschungsprojekte CollaBaWü (vgl. Kapitel 6.1) und WAVES (vgl. Kapitel 9.5) gegeben, bei denen sowohl Software-Unternehmen als auch Anwenderunternehmen beteiligt sind und die ein umfassendes Umfeld für Analysen in Software-Projekten zugänglich machen.

Somit ergibt sich folgende Einschätzung der Methoden:

- Mit einer Umfrage ist die Erhebung der Qualität der Anwenderbeiträge nur unzureichend erfassbar. Zudem müsste es Menschen bzw. Organisationen geben, die bereits Erfahrung mit OpenProposal gesammelt haben. Da OpenProposal neuartig ist und keine ähnlichen Methoden bzw. Annotationssysteme existieren, ist davon auszugehen dass eine Umfrage nicht geeignet ist.

- Für die Durchführung von Experimenten ist eine Nachbildung der Situation einer asynchronen Anwenderbeteiligung in einer Laborumgebung notwendig. Da eine asynchrone Anwenderbeteiligung zeitlich versetzt und im Nutzungskontext der Anwender erfolgen soll, sind die Variablen für ein Experiment zu asynchroner Anwenderbeteiligung nicht kontrollierbar. Außerdem ist die Belastung der Anwender in einem Experiment nicht nachzubilden.

- Für diese Arbeit kristallisiert sich die Fallstudie als geeignete Methode heraus, da sie die Möglichkeit zur empirischen Evaluation einer neuartigen Methode in einem realen Kontext über einen längeren Zeitraum bietet und die Anwendung der Bewertungskriterien aus Kapitel 2.9 vollständig zulässt. Zur Steigerung der Robustheit und Generalisierbarkeit der Fallstudien-Methode hat sich nach Yin [2003] die Durchführung mehrerer Fallstudien mit einer zusammenhängenden Fragestellung, sogenannter vergleichender Fallstudien, als empfehlenswert erwiesen.

Die Methode der vergleichenden Fallstudien stellt somit einen geeigneten Kompromiss zwischen der wissenschaftlichen Qualität der Ergebnisse und anwendungsnaher Forschung im industriellen Umfeld dar.

9.2 Zur Methode der vergleichenden Fallstudien

Dieses Unterkapitel befasst sich mit der Methode der vergleichenden Fallstudien. Hierzu wird eine Definition von Fallstudien aufgeführt, der Unterschied zwischen einer Einzelfallstudie und einer vergleichenden Fallstudien erläutert und anschließend die grundlegende Vorgehensweise in der vergleichenden Fallstudie, wie sie auch in der Evaluation der OpenProposal Methode angewendet wird, vorgestellt.

9.2.1 Fallstudienarbeit als Forschungsansatz

Fallstudien als wissenschaftlicher Forschungsansatz sind von der umgangssprachlichen Verwendung des Begriffs im Sinne von Anekdoten, Fallbeispielen, Storytelling oder Business Cases abzugrenzen. Sie werden wie folgt definiert [Yin, 2003, S.13]:

> *„1. A case study is an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident. (…)*
>
> *2. The case study inquiry copes with the technically distinctive situation in which there will be many more variables of interest than data points, and as one result relies on multiple sources of evidence, with data needing to converge in a triangulating fashion, and as another result benefits from the prior development of theoretical propositions to guide data collection and analysis".*

Mit anderen Worten: eine Fallstudie ist eine empirische Methode, mit der Phänomene in ihrem Kontext untersucht werden können. Dies ist gerade dann von Vorteil, wenn der Kontext bzw. die Rahmenbedingungen nicht klar vom Untersuchungsgegenstand getrennt werden können bzw. in die Analysen mit einbezogen werden müssen. Das bedeutet, dass in einer Fallstudie unterschiedliche Datenquellen und Methoden zur Datenerhebung herangezogen werden können, um eine Theorie aufzustellen und zu untersuchen. Eine Fallstudie ist somit nicht als reine Datenerhebungstechnik sondern als eine umfassende Methode zu betrachten, die alle Schritte von der Theoriebildung über die Datenerhebung bis zur Datenanalyse beinhaltet.

Die Definition einer Fallstudie von Yin [2003] unterscheidet sich von anderen Definitionen insofern, als sie eine höhere Flexibilität und Offenheit aufweist. Im Gegensatz zu Stake [1995] ist es nach Yin nicht notwendig, theorielos vorzugehen. Außerdem lässt die Sichtweise von Yin [2003] auch die Durchführung von quantitativen Erhebungen explizit zu, worüber sich andere Definitionen ausschweigen (vgl. [Borchhardt und Göthlich, 2007]).

Die besonderen Stärken einer Fallstudie sollen im Vergleich zu rein quantitativen Forschungsstrategien in der umfassenderen und dadurch besseren Abbildung der sozialen Wirklichkeit liegen. Sie bleibt nicht auf statische Momentaufnahmen beschränkt, sondern erlaubt es, Entwicklungen, Prozessabläufe und Ursache-Wirkungs-Zusammenhänge nachzuvollziehen sowie praktisch relevante, datenbasierte Aussagen zu treffen. Die Fallstudie kann in einem weiten Spektrum universell eingesetzt werden. Je nach Zwecksetzung der Studie und Inhalt der Forschungsfrage sind auf einen Theorietest oder die (Weiter-)Entwicklung von Theorien abzielende Arbeiten möglich. Fallstudien erlauben allerdings keinen statistischen Induktionsschluss auf eine Grundgesamtheit.

Während mit quantitativen Analysen v.a. das Ziel verfolgt wird, aus bestehenden Theorien abgeleitete Hypothesen zu testen und damit bestehendes Wissen zu spezifizieren, eignet sich der hier dargestellte Forschungsansatz der Fallstudie nach Borchhardt und Göthlich [2007, S. 46] besonders dann, „*wenn es darum geht, komplexe, bisher wenig erforschte Phänomene in einem breiten Zugang und vor dem Hintergrund ihrer Kontextbezogenheit zu betrachten*“. Die Erkenntnisgewinnung durch Fallstudien zielt dabei auf das Erschließen neuen Wissens, die Entwicklung von Erklärungsmodellen und die Ableitung von Hypothesen ab. Sie kann auch zur Prüfung von Hypothesen herangezogen werden, bei der nicht die statistische Generalisierbarkeit sondern die analytische Generalisierung zur Weiterentwicklung von Theorien im Vordergrund steht.

9.2.2 Einzelfallstudie im Vergleich zur vergleichenden Fallstudie

Hinsichtlich der Fallauswahl unterschiedet Yin [2003] zwei Typen von Fallstudien: die Einzelfallstudie („single-case design“) und die vergleichende Fallstudie („multiple-case design“). Im Gegensatz zur Einzelfallstudie untersucht die vergleichende Fallstudie mehrere Fälle, vergleicht sie miteinander und zieht fallübergreifende Schlussfolgerungen. Ein Fall kann sich bei beiden Fallstudientypen auf eine gesamte Untersuchungseinheit („holistic“, z.B. ein Markt, ein Projekt, ein Unternehmen, eine Abteilung) oder auf mehrere Untersuchungseinheiten („embedded“, z.B. mehrere Projekte, mehrere Unternehmen) beziehen.

Die Einzelfallstudie konzentriert sich zumeist auf kritische, extreme, einzigartige, repräsentative, typische oder bisher nicht zugängliche Fälle oder solche, die über einen längeren Zeitraum beobach-

tet werden. Einzelfallstudien werden z.B. durchgeführt, um theoretische Erkenntnisse in Frage zu stellen oder neue Erkenntnisse in Bezug auf bislang unerforschte Phänomene zu gewinnen.

Der Vorteil einer vergleichenden Fallstudie gegenüber einer Einzelfallstudie liegt darin, dass die gewonnenen Erkenntnisse durch Ähnlichkeiten und Unterschiede zwischen den Fällen kritisch beleuchtet werden können. Aus diesem Grund gelten die Ergebnisse vergleichender Fallstudien als überzeugender, vertrauenswürdiger und robuster (vgl. [Eisenhardt, 1989, S. 541; Miles und Huberman, 1994, S. 29; Yin, 2003, S. 19, S. 53]). Dem steht entgegen, dass mit dem Forschungsansatz der vergleichenden Fallstudie, abgesehen von den hohen Kosten, vor allem ein erheblicher Zeitaufwand verbunden ist. Die zu untersuchenden Fälle haben in einem Zusammenhang mit dem Forschungsziel zu stehen, dürfen in diesem Rahmen aber durchaus beliebig, wenngleich auch begründet ausgewählt werden (vgl. [Eisenhardt, 1989, S.537; Stake, 1995, S. 4]), um bewusst bestimmte Typen von Fällen zu erfassen. Die Fallauswahl hat anders als in der quantitativen Forschung keinem Zufallsprinzip zu gehorchen.

Ähnlich einer Serie von Experimenten in den Naturwissenschaften folgen vergleichende Fallstudien einer Replikationslogik („replication logic“). Weitere Fälle werden dabei so ausgewählt, dass sie den Rahmenbedingungen der ersten analysierten Fälle entsprechen (vgl. [Miles und Huberman, 1994, S. 29; Yin, 2003, S. 47]), sodass sie voraussichtlich die bisherigen Erkenntnisse bestätigen („literal replication“). Alternativ können Fälle selektiert werden, bei denen anders lautende Resultate erzielt werden, die aber theorieseitig vorhersagbar sind („theoretical replication“).

Richtgröße für eine vergleichende Fallstudie ist eine Anzahl von vier bis zehn Fällen, da sich bei einer größeren Zahl die Komplexität der Auswertung erheblich erhöht (vgl. [Eisenhardt, 1989, S. 545]) und sich bei einer geringeren Zahl die Aussagekraft der Fälle verringert. Auch zwei bis drei Fälle sind nach Yin [2003, S. 47] zulässig, wenn ähnliche Fälle untersucht werden und die Replikation der Erkenntnisse im Vordergrund der Untersuchungen steht.

9.2.3 Vorgehensweise in der vergleichenden Fallstudie

Die Durchführung einer vergleichenden Fallstudie erfolgt nach dem in Abbildung 9.1 illustrierten Ablaufschema von Borchhardt und Göthlich [2007, S. 44], welche sich auch mit den Empfehlungen von Yin [2003], der COSMOS Gruppe [2000] und Stake [2006] deckt. Nach der Planung des Forschungsprozesses wird für jede Fallstudie mit der Datenerhebung und Auswertung begonnen. Nach Abschluss jeder einzelnen Fallstudie werden die Teilnehmer gebeten, die Auswertungen in Form eines Fallstudienberichts auf Korrektheit und Vollständigkeit zu überprüfen und die kommunikative Validität der Auswertungen sicherzustellen. Zur argumentativen Validierung bietet es sich darüber hinaus an, die Fallstudienberichte auch externen Personen zur kritischen Betrachtung zur Verfügung zu stellen.

Im Anschluss folgt die Interpretation der einzelnen Fälle und im Rahmen der fallvergleichenden Analyse die qualitative Inhaltsanalyse und fallstudienübergreifende Interpretation der Ergebnisse. Handelt es sich um eine hypothesenprüfende Fallstudienarbeit, wird die Gültigkeit der aufgestellten Hypothesen anhand der Ergebnisse geprüft. Für eine hypothesengenerierende Fallstudienarbeit kann mit den Ergebnissen die Erklärung der untersuchten Phänomene erarbeitet werden. Hierzu werden Ursache-Wirkungs-Ketten untersucht, logische Modelle entwickelt und schließlich Hypothesen abgeleitet. Wurde eine Langzeitstudie durchgeführt, lassen sich chronologische Entwicklungen beschrei-

ben. Die Form, Struktur und Darstellung des Berichts ist frei wählbar. Zu beachten ist, dass eine direkte Vergleichbarkeit der einzelnen Fallstudien möglich ist, und dass zum Schutz der untersuchten Personen bzw. Unternehmen eine Anonymisierung der Ergebnisse erfolgt.

Im Folgenden werden die Empfehlungen zur Planung des Forschungsprozesses, zur Datenerhebung und zur Auswertung erläutert. Im Abschnitt Auswertung werden die Grundlagen zur Auswertung der einzelnen Fälle, zur kommunikativen Validierung, zur Interpretation der einzelnen Fälle, zur fallvergleichenden Analyse und Interpretation und zum Bericht zusammengefasst betrachtet.

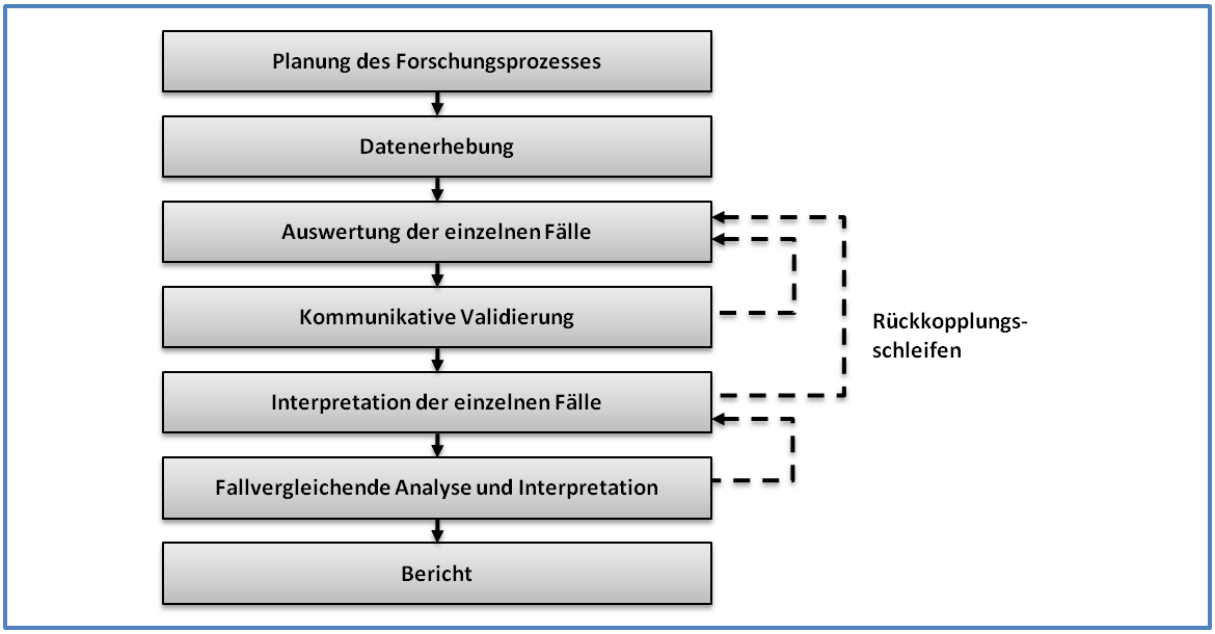

Abbildung 9.1: Ablauf einer vergleichenden Fallstudie [Borchhardt und Göthlich, 2007]

Planung des Forschungsprozesses

Ziel des Planungsprozesses in Fallstudienarbeiten ist die Entwicklung und Niederschrift eines Forschungsprotokolls, das den weiteren Fortgang der Untersuchung leitet. Darin sind die Problemstellung und Zielsetzung der Analyse, eine Definition und Auswahl der Fälle sowie die anzuwendenden Datenerhebungsmethoden festzulegen (vgl. [Mayring, 2002, S. 43f]). Einerlei, ob das Ziel der Arbeit im Aufstellen oder Testen von Hypothesen besteht, ist nach der hier verfolgten Auffassung ein theoriegeleitetes Vorgehen unter Berücksichtigung der relevanten Literatur zu wählen. Davon ausgehend sind Hypothesen oder zumindest pragmatische Aussagesysteme und Vermutungen zu generieren, die es im Verlauf der Studie zu untersuchen bzw. denen es zu folgen gilt (vgl. [Yin, 2003, S.9, S.28f]).

Dabei bilden nach Yin [2003, S. 21ff] die folgenden fünf Komponenten die zentralen Grundpfeiler jeder Fallstudie:

- Die Forschungsfrage („the research question“);
- theoretische Propositionen bzw. Hypothesen („the study propositions“);
- der Gegenstand der Untersuchungen („the unit of analysis“);
- die Verknüpfungslogik zwischen Daten und Hypothesen („the logic of linking the data to the propositions“);

- Kriterien zur Interpretation der Ergebnisse („the criteria for interpreting the findings“).

Die Forschungsfrage beinhaltet die Zielsetzung der Analyse. Eine Forschungsfrage kann nach Yin [2003] einer der Kategorien „Wer“, „Was“, Wo“, „Wie“ und „Warum“ angehören. Für Fallstudien empfiehlt Yin [2003] die Formulierung von Fragen der Form „Wie“ („How?“) oder „Warum“ („Why?“). Nach Kitchenham et al. [1995] sollten sich Fallstudien in der Domäne der Software-Technik jedoch nicht nur mit dem „Wie“ und „Warum“ sondern auch mit der Frage „Was ist besser“ („Which is better?“) beschäftigen, um Methoden und Werkzeuge in Fallstudien zu evaluieren. Die anschließende Bildung von Hypothesen bzw. Grundannahmen dient der Fokussierung der Untersuchungen. Sie stellt eine Konkretisierung der Forschungsfrage dar und reduziert die Komplexität einer Untersuchung auf die gestellten Hypothesen. Der Gegenstand der Untersuchungen definiert, worauf sich der Fall bezieht. Er leitet sich aus der Forschungsfrage und den Hypothesen ab und kann sich von einem einzelnen Individuum auf beliebige organisatorische Einheiten erstrecken und auch zeitliche Ereignisse umfassen. Letztendlich stellt sie eine weitere Reduzierung der Komplexität der Untersuchungen dar. Die Verknüpfung der erhobenen Daten mit den zuvor aufgestellten Hypothesen ist fallspezifisch zu bestimmen. Wenn möglich sollten auch Kriterien zur Bewertung der Ergebnisse festgelegt werden. Damit soll definiert werden, welche Bedeutung die gewonnenen Ergebnisse besitzen und welche Aussagen darüber getroffen werden können.

Zur Sicherung der Qualität des Forschungsprozesses postulieren Yin [2003], Lamnek [2005] und Borchhardt und Göthlich [2007] die Kriterien Konstruktvalidität, interne und externe Validität, Reliabilität und Objektivität. Mit ihnen soll eine Qualitätssicherung in der Konstruktion des Forschungsansatzes, in der Datenerhebung und in der Auswertung der Daten gewährleistet werden.

Die Konstruktvalidität bezeichnet die Gültigkeit bzw. die Zulässigkeit der Operationalisierung des Konstrukts, die dem Forschungsprozess zugrunde liegt. Sie geht der Frage nach, ob im Forschungsprozess tatsächlich gemessen wird, was zu messen beabsichtigt wird, und ob anhand der Messung Aussagen über das Konstrukt getroffen werden dürfen. Als Taktiken zur Absicherung der Konstruktvalidität wird empfohlen, erstens mehrere Erhebungsverfahren zur Betrachtung des Untersuchungsgegenstands aus unterschiedlichen Perspektiven (Methodentriangulation) zu verwenden und deren Durchführung nachvollziehbar und überprüfbar zu dokumentieren, zweitens eine a-priori Spezifikation der interessierenden Konstrukte aufbauend auf die relevante Literatur vorzunehmen und drittens die kommunikative Validierung sorgfältig durchzuführen.

Die interne Validität beschreibt die Gültigkeit der aufgestellten Kausalzusammenhänge, ihre intersubjektive Überprüfbarkeit und die Zuverlässigkeit. Zur Steigerung der internen Validität bietet es sich an, vergleichende Fallstudien durchzuführen sowie den Interpretationsprozess und die daraus gewonnenen Erkenntnisse in einer argumentativen Validierung zu explizieren und überprüfbar bereitzustellen.

Die Generalisierbarkeit der Untersuchungsergebnisse wird als externe Validität bezeichnet. In quantitativen Analysen mit einer großen repräsentativen Stichprobe kann eine statistische Generalisierbarkeit sichergestellt werden. Bei qualitativen Analysen mit einer vergleichsweise geringen Anzahl an Stichproben ist eine statistische Generalisierbarkeit nicht zulässig. Stattdessen ist eine analytische Generalisierbarkeit anzustreben, bei der die gewonnen Erkenntnisse analytisch zu einer Theorie verallgemeinert werden (vgl. [Yin, 2003, S. 37f]).

Die Reliabilität einer Fallstudie ist gegeben, wenn eine Wiederholung der Fallstudie mit identischen Voraussetzungen und der gleichen Vorgehensweise die gleichen Ergebnisse und Schlussfolgerungen erzielen wird. Hierfür ist eine präzise Dokumentation der Untersuchungen notwendig. Bei qualitativen Forschungsmethoden wird allerdings im Allgemeinen bezweifelt, dass die identischen Bedingungen vorausgesetzt werden können (vgl. [Borchhardt und Göthlich, 2007]).

Die Absicherung der Objektivität erfolgt bei quantitativen Untersuchungsmethoden durch eine Standardisierung der Datenerhebung und –auswertung. Bei einem qualitativen Untersuchungsansatz ist die Objektivität nur dann gegeben, wenn eine intersubjektive Nachprüfbarkeit durch die Explikation des Untersuchungsablaufs und des Interpretationsprozesses sichergestellt wird.

Datenerhebung

Die Datenerhebung im Rahmen von Fallstudienarbeiten ist an keine bestimmte Erhebungsmethode gebunden. Fallstudien sind vielmehr durch eine Methodentriangulation gekennzeichnet, bei der die unterschiedlichen Perspektiven der Fallstudienteilnehmer auf den Untersuchungsgegenstand mit Hilfe der verschiedenen Erhebungsmethoden erfasst werden. Die Wahl der Erhebungsmethoden erfolgt dabei abhängig von den Untersuchungssituationen und –objekten.

Zu den etablierten Erhebungsmethoden gehören die Befragung (vgl. [Böhler, 2004]), die Beobachtung (vgl. [Atteslander, 2003; Bortz und Döring, 2003]), das Experiment (vgl. [Erichson, 1995]), das Panel (vgl. [Hüttner, 2002]), die Dokumenten- und Inhaltsanalyse (vgl. [Mayring, 2007]) und die Inventur (vgl. [Krallmann und Frank, 2002]). Die Auswahl der Erhebungsmethode sollte dem zu untersuchenden Forschungsgegenstand angemessen sein und sich nach der Zielsetzung der Untersuchung richten. Sie sollte nach Kaya [2007, S. 51] u.a. abhängig von dem Untersuchungsvorhaben, der Zielgruppe, der erforderlichen Informationsqualität und den Zeit- und Kostenrestriktionen getroffen werden. Da in den Fallstudien dieser Arbeit die Methoden der Befragung, der Beobachtung und der Dokumenten- und Inhaltsanalyse Anwendung finden, werden diese im Folgenden näher erläutert.

In der Befragung werden Personen mit Hilfe von Stimuli (z.B. schriftlicher Fragebogen, Bilder, mündliche Beschreibung) bezüglich ihrer Aussage bzw. Meinung befragt (vgl. [Böhler, 2004]). Es wird zwischen persönlichen und telefonischen sowie schriftlichen Befragungen unterschieden. Eine Sonderform stellen Fokusgruppen bzw. Gruppendiskussionen dar. Jede einzelne Form der Befragung besitzt Vor- und Nachteile (vgl. [Aaker und Day, 2003; Böhler, 2004; Bohnsack und Przyborski, 2007; Hammann und Erichson, 2006; Hüttner, 2002]), die bei der Auswahl der Befragungsform berücksichtigt werden sollten.

Die persönliche Befragung lässt eine hohe Flexibilität während der Befragung zu und sichert die höchste Antwortquote zu, da im Gespräch mit der befragten Person bei unvollständigen Antworten nachgefasst und zu einer hilfreichen Antwort motiviert werden kann. Allerdings hat sie den Nachteil, dass die befragte Person im Interview möglicherweise beeinflusst wird und damit Verzerrungen der Antworten auftreten können. Darüber hinaus ist mit einem hohen Aufwand zur Durchführung der Befragungen zu rechnen. Die telefonische Befragung ist sehr einfach und schnell durchführbar und ermöglicht die Kontaktaufnahme zu vielen Personen. Nachteil der telefonischen Befragung ist, dass sie keine Stimuli zulässt, dass am Telefon nur wenige Fragen gestellt werden können und wie auch bei der persönlichen Befragung die Gefahr einer Beeinflussung der befragten Person besteht. Die Vorteile der schriftlichen Befragung liegen darin, dass eine Beeinflussung bei der Befragung nahezu

ausgeschlossen werden kann. Ferner ist der Zeitaufwand für die Erhebung der Antworten gering. Sie besitzt jedoch den Nachteil, dass die Erstellung der Befragung eine aufwändige Vorbereitung voraussetzt. Ferner ist die Flexibilität der Befragung gering, da der zu erfragende Themenbereich und der Fragenumfang sehr begrenzt sind. Fokusgruppen bzw. Gruppendiskussionen (vgl. [Bohnsack und Przyborski, 2007]) und die damit einhergehende Interaktion der Teilnehmer miteinander bietet den Vorteil, dass sich die Teilnehmer mit ihren Aussagen gegenseitig inspirieren und Themen auf diese Weise vielseitiger und umfassender diskutiert werden können als in Einzelinterviews. Mit ihrer Hilfe könnten unerwartete und zuvor unbedachte Aspekte und Zusammenhänge aufgedeckt werden. Nachteile der Fokusgruppen liegen in ihrem hohen zeitlichen Aufwand und ihrer eher explorativen Natur, so dass ihre Ergebnisse hauptsächlich aus Meinungen und Vorstellungen bestehen, die gründlich gedeutet werden müssen.

Um eine sorgfältige und nachvollziehbare Datenerhebung sicherzustellen existieren für die Vorbereitung, Durchführung und Nachbereitung von Befragungen zahlreiche Leitlinien (vgl. [Converse und Presser, 1986; Dillmann, 1978; Payne, 1951; Schnell et al., 2008]). Abhängig vom Ziel der Untersuchung kann der Strukturierungsgrad der Befragung variiert werden. Während eine offene, weniger strukturierte Methode wie z.B. das narrative oder problemzentrierte Interview für eine komplexe und neuartige Fragestellung geeignet ist, kann in anderen Fällen eine stärker strukturierte Befragung wie z.B. das Experteninterview, das halbstandardisierte Interview oder das fokussierte Interview sinnvoll sein. Zusätzlich kann bei der schriftlichen Befragung zwischen einer quantitativen und einer qualitativen Ausrichtung gewählt werden. Dabei ist vor allem auf die Form, Verständlichkeit und die Reihenfolge der Fragestellung sowie auf formale Kriterien bei der Durchführung der Befragung zu achten. Fragen sollten u.a. konkret, einfach, neutral und nicht hypothetisch formuliert sein, sich nur auf den Sachverhalt beziehen, keine doppelten Negationen enthalten, formal balanciert sein und keine bestimmte Beantwortung provozieren. Für eine höhere Akzeptanz der Befragten sollte der Testleiter die Befragung mit einleitenden Worten beginnen, und zuerst einfache und leicht zu beantwortende Fragen und später die schwierigeren Fragen stellen. Vor dem Start der Befragungen sollten Tests bzw. Pretests durchgeführt werden, um auf mögliche Fehlerquellen zu prüfen.

Die Beobachtung kann sehr unterschiedliche Formen annehmen. Es wird zwischen teilnehmend und nicht-teilnehmend, intern und extern, verdeckt und offen, informiert und unwissentlich, wenig und stark strukturiert sowie direkt und indirekt unterschieden (vgl. [Gehrau, 2002, S. 28ff; Schnell et al., 2008]). Außerdem kann unterschieden werden, ob das eigene Verhalten oder das von anderen beobachtet wird (vgl. [Brosius et al., 2009; Gehrau, 2002]). Im Gegenzug zur nicht teilnehmenden Beobachtung sieht der Forscher bei der teilnehmenden Beobachtung nicht nur zu, sondern übernimmt eine Rolle im untersuchten Prozess bzw. in der untersuchten Umgebung, um so eigene Erfahrungen zu sammeln. Der Forscher bzw. der Beobachter kann dabei zur Gruppe der beobachteten Personen gehören (intern) oder extern beauftragt sein. Bei der offenen Beobachtung ist der Forscher für alle beobachteten Personen sichtbar. Dies kann allerdings eine Beeinflussung der beobachteten Personen bzw. Gruppen bewirken, den sogenannten Beobachtungseffekt. In einer verdeckten Beobachtung gibt sich der Beobachter durch Tarnung oder technische Hilfsmittel nicht zu erkennen. Eine Beeinflussung findet zwar nicht statt, allerdings sollte diese Vorgehensweise aus forschungsethischen Gründen nur in begründeten Fällen und nur in einem ethisch vertretbaren Umfang angewendet werden. Üblicherweise werden die beobachteten Personen vor Beginn einer Untersuchung über Vorgehensweise, Sinn und Hintergrund der Studie informiert. In bestimmten Fällen kann es jedoch

sinnvoll sein, dass eine Beobachtung bewusst unwissentlich erfolgt. Eine stark strukturierte Beobachtung ist durch eine klare Definition der zu untersuchenden Phänomene und Erfassungskategorien gekennzeichnet. Für Vorstudien und explorative Studien kann allerdings auch eine weniger strukturierte Beobachtung vorgenommen werden. Während bei einer direkten Beobachtung ein Geschehen unmittelbar zu dem Zeitpunkt erfasst wird, an dem es passiert, werden bei einer indirekten Beobachtung Schlussfolgerungen aus bestimmten bereits vorliegenden Fakten bzw. Tatsachen gezogen, z.B. bei einer Qualitätskontrolle.

Die Sammlung und Analyse von Dokumenten stellt die dritte Erhebungsmethode dar. Neben unterschiedlichen Schriftstücken zählen nach Yin [2003] hierzu auch Archivdatensätze, d.h. Datenquellen, die zusätzlich über quantitative und/oder nicht-textliche Informationen verfügen (z.B. Bilanzen, Organigramme, Besprechungsprotokolle, etc.). Die Methodik der Inhaltsanalyse beinhaltet dabei nach Mayring [2007, S. 58ff] die drei Grundschritte Zusammenfassung, Explikation bzw. Erklärung und Strukturierung. Nach einer überblicksartigen Erfassung der Inhalte und Reduktion auf Kernaussagen (Zusammenfassung) werden in der Explikation einzelne Passagen der Quelle mit Bezug auf zusätzliche externe Informationen erklärt und damit zeitlich und sachlich geordnet. Anschließend findet mit der Strukturierung eine Kategorisierung innerhalb der Quelle statt, bei der sich das Kategoriensystem aus dem Forschungsprotokoll ableiten lässt.

Auswertung

Die Auswertung bzw. Analyse stellt nach Yin [2003] die anspruchsvollste Phase einer Fallstudie dar, da die Strategien und Methoden zur Analyse nicht eindeutig definierbar sind. Borchardt und Göthlich [2007] resümieren, dass sich die Datenanalyse im Rahmen vergleichender Fallstudien als komplex und schwierig gestaltet, da in der Literatur keine klaren Handlungsanweisungen und keine Verfahrensweisen vorgegeben werden. Die „eine richtige Fallstudienanalyse“ gibt es nicht. Stattdessen existieren zahlreiche Werkzeuge zur Planung und Durchführung der Auswertung bzw. Analyse.

Die Auswertung gliedert sich grundsätzlich in die Analyse und Interpretation der einzelnen Fälle (Einzelfallanalyse, „within-case analysis“) und deren kommunikative Validierung sowie die anschließende fallübergreifende Datenanalyse („cross-case analysis“) und Interpretation, die zu einem Bericht zusammengefasst werden. Miles und Hubermann [1994] beschreiben hierzu mehrere Analysetechniken für die Datenanalyse. Sie empfehlen eine Strukturierung, Verdichtung und grafische Visualisierung der Informationen, um die Daten aus unterschiedlichen Betrachtungswinkeln zu untersuchen, die Komplexität zu reduzieren, Muster zu erkennen und Interpretationen abzuleiten.

Yin [2003] weist darauf hin, dass die Manipulation der Daten und das „Spielen mit den Daten“ zwar hilfreich sein kann, aber erst mit einer übergeordneten Analysestrategie eine effiziente und erfolgsversprechende Vorgehensweise gewählt wird. Er nennt drei generelle Strategien, die in Erwägung gezogen werden sollten. Erstens sollte sich die Analyse an den aufgestellten Hypothesen bzw. theoretischen Propositionen orientieren. Dabei sollte ein Bezug zwischen den Daten und den Hypothesen hergestellt werden. Zweitens sollten rivalisierende Erklärungsmodelle bzw. Hypothesen definiert und überprüft werden, falls diese nicht bereits in der initialen Aufstellung der Hypothesen bzw. theoretischen Propositionen enthalten sind. Die dritte Strategie verfolgt die Entwicklung eines deskriptiven Rahmenwerks zur Organisation der Fallstudie. Sie sollte nur gewählt werden, wenn die anderen beiden Strategien nicht anwendbar sind und die Fallstudie auf eine deskriptive Vorgehensweise ausgerichtet ist.

Als Analysetechniken nennt Yin [2003] fünf Techniken: Mustervergleich („Pattern Matching“), Erklärungsmodell („Explanation Building“), Zeitreihenanalyse („Time-Series Analysis“), Logikmodelle („Logic Models“) und fallübergreifende Synthese („Cross-Case Synthesis“).

Der Mustervergleich stellt die am häufigsten eingesetzte Technik der Fallstudienanalyse zur Erhöhung der internen Validität dar. Der Forscher erstellt zu Beginn eines oder mehrere Muster auf und untersucht, ob und wie sie mit dem Muster, das im Rahmen der Fallstudie beobachtet wurde, übereinstimmen. Muster können abhängige bzw. unabhängige Variablen einer Untersuchung („nonequivalent dependent variables as patterns“, vgl. [Yin, 2003, S. 116]) bzw. „simpler patterns“, vgl. [Yin, 2003, S. 119]) oder rivalisierende Erklärungen zu einer Fragestellung („Rival explanations as patterns“, vgl. [Yin, 2003, S. 118]) beschreiben. Der Grad der Exaktheit der Übereinstimmung zwischen dem vorhergesagten und dem tatsächlich beobachteten Muster ist dabei nach den Möglichkeiten der Messung und Datenerhebung in der Fallstudie zu wählen. Hierfür empfiehlt Yin [2003] die Entwicklung präziser Kennzahlen bzw. Metriken und einfacher, klarer Muster, damit bei der Interpretation der Ergebnisse eine eindeutige Aussage über die Korrektheit des vorhergesagten Musters getroffen werden kann.

Eine Sonderform des Mustervergleichs ist die Entwicklung von Erklärungsmodellen bzw. Ursache-Wirkungs-Ketten zu einer Fallstudie. Dabei werden Erklärungen zu Phänomenen abgegeben, die mit einer Fallstudie verifiziert oder auch falsifiziert werden. Die Entwicklung der Erklärungsmodelle kann zudem auch iterativ erfolgen, so dass die Erklärungen überprüft, angepasst und wiederum überprüft werden. Für eine gute Fallstudie empfiehlt Yin [2003], dass sich die Erklärungsmodelle auf die in der Fallstudie aufgestellten theoretischen Propositionen bzw. Hypothesen beziehen. Abhängig davon ob Propositionen bzw. Hypothesen gültig sind, werden in den Erklärungsmodellen die daraus entstehenden Konsequenzen bzw. Schlussfolgerungen erklärt.

Die dritte Technik stellt die Zeitreihenanalyse dar, mit der Erhebungsdaten über einen bestimmten Zeitraum erfasst und auf zeitliche Aspekte bezogene Auswertungen erstellt bzw. Schlussfolgerungen gezogen werden. Dabei sollen über bestimmte Zeitintervalle Trends identifiziert, ein Vergleich mit vorher auftretenden Trends durchgeführt und eine Prognose über mögliche zukünftige Zeitreihenwerte abgegeben werden können. Während in einfachen Zeitreihenanalysen nur eine abhängige bzw. unabhängige Variable betrachtet wird, werden in komplexen Zeitreihenanalysen mehrere Variablen gleichzeitig erfasst und deren Veränderungen über die Zeit berücksichtigt. Außerdem kann eine Zeitreihenanalyse zur Analyse chronologischer Ereignisse herangezogen werden, um zeitliche Abhängigkeiten identifizieren zu können.

Für Fallstudien mit dem Ziel einer Evaluation hat sich in den letzten Jahren die Technik der Logikmodelle als zweckmäßig erwiesen. In einem Logikmodell wird aufbauend auf empirischen Untersuchungen ein Ursache-Wirkungs-Diagramm entwickelt. Eine abhängige Variable beschreibt ein Ereignis zum Zeitpunkt t und wird zum Zeitpunkt t+1 in eine unabhängige Variable transformiert, die den ausgelösten Effekt beschreibt. Im Unterschied zum Mustervergleich beziehen sich Logikmodelle auf eine Sequenz von ereignisbasierten Ursache-Wirkungs-Modellen. Darüber hinaus können mit diesem Modell auch rivalisierende Ursache-Wirkungs-Modelle beschrieben und untersucht werden. Es unterscheidet dabei vier Ebenen: die Ebene eines Individuums („Individual-level logic model“), die Ebene einer Organisation („Firm or organizational-level logic model“), die Ebene der alternativen Konfigurationen einer Organisation („An alternative configuration for an organizational-level logic

model“) und die Ebene eines politischen Programms („Program-level logic model“). Auf Ebene des Individuums bezieht sich ein Logikmodell auf das Verhalten bzw. den Zustand einer Person bzw. einer bestimmten Gruppe von Personen. Auf der Ebene der Organisation beschreibt ein Logikmodell die Abläufe einer einzelnen Organisation bzw. einer Kategorie von Organisationen. Möchte man zudem die dynamischen Veränderungsprozesse einer Organisation berücksichtigen, verwendet man Logikmodelle mit alternativen Konfigurationen einer Organisation. Darin unterscheidet ein Logikmodell zwischen mehreren Konfigurationen einer Organisation und beschreibt für jede Konfiguration ein spezifisches Modell. Auf Ebene eines politischen Programms kann ein Logikmodell ausgehend von der Situation und den Maßnahmen mögliche Effekte beschreiben.

Die fünfte Technik, die fallübergreifende Synthese, kann bei fallvergleichenden Fallstudien angewendet werden. Die Fälle werden zuerst unabhängig voneinander als separate Fälle betrachtet und ausgewertet. Mit Hilfe einer Datentabelle („word table“) wird eine übergeordnete Datenstruktur aufgestellt, mit der die relevanten Aspekte der einzelnen Fälle zusammengefasst nebeneinander gestellt und fallstudienübergreifend analysiert werden können. Die Ziele dieser Datentabellen sind die Sicherstellung der Vergleichbarkeit der Fälle, die fallvergleichende Herausarbeitung der relevanten Fakten aus den Berichten zu den einzelnen Fällen und die Erleichterung der Argumentation der Interpretationen und Schlussfolgerungen. Yin [2003] betont, dass die fallübergreifende Synthese vor allem auf eine argumentative Interpretation und nicht auf numerischen Berechnungen beruht. Sie verhält sich somit analog zu Experimenten mit einer kleinen Anzahl von Stichproben.

9.3 Planung der vergleichenden Fallstudien

Im Folgenden wird die Planung des Forschungsprozesses (vgl. [Borchhardt und Göthlich, 2007; Yin, 2003]) gemäß den vergleichenden Fallstudien beschrieben. Hierzu werden nach Yin [2003] die Forschungsfragen, die Hypothesen der Untersuchungen, der Untersuchungsgegenstand, die Verknüpfungslogik zwischen Daten und Hypothesen und die Kriterien zur Interpretation der Ergebnisse fallübergreifend bestimmt. Im Anschluss wird mit der Durchführung jeder einzelnen Fallstudie begonnen und schließlicheine fallübergreifende Diskussion und Interpretation der Ergebnisse vorgenommen.

9.3.1 Forschungsfragen der Fallstudien

Der Evaluation der OpenProposal Methode liegt die Frage zugrunde, ob mit OpenProposal eine effiziente und effektive Anwenderbeteiligung möglich ist. Die Analyse der Potentiale von Methoden zur asynchronen Anwenderbeteiligung in Kapitel 4 ergab, dass vor allem die Qualität der Anwenderbeiträge und die Belastung der Anwender Schwachstellen von asynchronen Methoden darstellen. In den Fallstudien ist daher gemäß den Bewertungskriterien aus Kapitel 2.9 ist zu prüfen, welche Effekte OpenProposal auf die Effizienz der Entwicklung, die Qualität der Anwenderbeiträge und die Belastung der Anwender hat.

Die Frage nach der Gebrauchstauglichkeit von OpenProposal führt zwangsläufig zur Frage nach der Akzeptanz von Seiten der Anwender und der Moderatoren. Um OpenProposal erfolgreich in einer praxisnahen Umgebung einsetzen zu können, ist eine positive Wahrnehmung der Methode aus dem Blickwinkel der Anwender von großer Bedeutung. Für die Evaluation stellt sich hierbei die Frage, ob die in der Anforderungsanalyse erfassten Anforderungen (vgl. Kapitel 6.3, Seite 89ff) zutreffen und

mit der Konzeption und Implementierung von OpenProposal korrekt umgesetzt wurden. Außerdem kann OpenProposal nur als sinnvoll bewertet werden, wenn es den Moderator in seinem Aufgabenbereich unterstützt, denn üblicherweise wird die Auswahl der Methoden zur Anwenderbeteiligung vom Moderator bestimmt. Daher stellt sich die Frage, ob und wenn ja warum OpenProposal zur Erstellung von Anwenderbeiträgen bei Anwendern und Moderatoren auf Akzeptanz stößt. Forschungsfrage F1 lautet somit:

F1: Wird OpenProposal von Anwendern und Moderatoren zur Erstellung von Anwenderbeiträgen in Software-Projekten akzeptiert und wenn ja weshalb?

Neben der Unterstützung der Erstellung der Anwenderbeiträge stellt sich die Frage, ob die mit OpenProposal erstellten Anwenderbeiträge bei der weiteren Verarbeitung als hilfreich oder als störend wahrgenommen werden. Dies betrifft vor allem den Moderator, der die Anwenderbeiträge der Anwender strukturieren und an Entscheider und Entwickler kommunizieren muss. Auch von Seiten der Entscheider und Entwickler muss OpenProposal akzeptiert werden, da nur in diesem Fall die Zusammenarbeit mit den Anwendern kooperativ und motivierend verlaufen kann. Werden die mit OpenProposal erstellten Anwenderbeiträge nicht wertgeschätzt, wird sich dies auch auf die Nutzung von OpenProposal auswirken. Forschungsfrage F2 lautet somit:

F2: Wird OpenProposal von Moderatoren, Entscheidern und Entwicklern in Software-Projekten akzeptiert und wenn ja weshalb?

Eine hohe Akzeptanz einer Methode stellt jedoch nicht unmittelbar sicher, dass die Methode die einzige Alternative zu einer Verbesserung der Prozesse darstellt. Daher soll in den Fallstudien auch der Frage nach einem Vergleich von OpenProposal mit anderen Methoden nachgegangen und die Gültigkeit der Analyse der existierenden Methoden (vgl. Kapitel 3 und Kapitel 4) geprüft werden. Forschungsfrage F3 lautet somit:

F3: Welche Vor- und Nachteile hat OpenProposal gegenüber anderen Methoden?

9.3.2 Hypothesen und Metriken der Fallstudien

Für die Fallstudien lassen sich aufbauend auf den Forschungsfragen fünf Hypothesen bilden, anhand derer die zu erwartenden Verbesserungseffekte von OpenProposal in den Fallstudien geprüft werden sollen.

Zur Evaluation einer Methode in Fallstudien ist die Verwendung von Metriken empfehlenswert (vgl. [Kitchenham und Pickard, 1995; Stake, 1995; Stake, 2006; Yin, 2003]). Dabei unterscheidet man zwischen unabhängigen und abhängigen Variablen. Eine Variable ist unabhängig, wenn sie in einer Untersuchung variiert werden kann, um ihre Auswirkungen auf eine abhängige Variable zu erfassen. Somit stellen die unabhängigen Variablen die unterschiedlichen Variationen einer Untersuchung und die abhängigen Variablen die Metriken der Untersuchungen dar. Damit ist festzustellen, welche Effekte die unterschiedliche Ausprägungen einer unabhängigen Variable auf die abhängigen Variablen haben.

Im Folgenden wird die Bildung dieser Hypothesen und der jeweils zugehörigen Metriken erläutert.

Die Analyse der Potentiale der existierenden Methoden aus Kapitel 3 und 4 hat ergeben, dass die bisherigen Methoden Nachteile besitzen, die die Effizienz und Effektivität der Anwenderbeteiligung mindern könnten. Bei ihnen mangelt es vor allem an einer Unterstützung des Anwenders bei der Erstellung von Anwenderbeiträgen, durch die auch der Aufwand des Moderators für die Vor- und Nachbereitung reduziert wird. Aus Sicht der Entscheider und Entwickler sind vor allem Mängel in der Verständlichkeit der Anwenderbeiträge und der hohe zeitliche Aufwand zur Erfassung der Botschaften der Anwenderbeiträge problematisch. Mit der Konzeption und Implementierung von OpenProposal wird der Anspruch erhoben, diese Mängel zu beseitigen – und auf diese Weise auf Akzeptanz von Seiten aller Beteiligten zu stoßen – sowie Vorteile gegenüber den bisherigen Methoden aufzuweisen.

Es wird somit erwartet, dass der Anwender Anwenderbeiträge mithilfe von OpenProposal leichter und schneller erstellen kann. Dies bedeutet, dass mit OpenProposal im Vergleich zu anderen Methoden in der gleichen Zeit mehr Anwenderbeiträge erstellt werden sollten. Hypothese H1 lautet somit:

H1: Mit OpenProposal werden im Vergleich zu anderen Methoden mehr Anwenderbeiträge erfasst.

Es wird erwartet, dass die Anwender durch die Vereinfachung der Erstellung von Anwenderbeiträgen sowie durch den direkten Bezug zur Anwendungsoberfläche mithilfe des OpenProposal Annotationssystems motiviert werden, ihre Vorstellungen und Wünsche genauer zu erläutern und nicht nur kosmetische Verbesserungen der Anwendungsoberfläche sondern auch wichtige konstruktive Anforderungen bzw. wichtige Akzeptanzfaktoren („critical incident reports“) mitzuteilen. Hypothese H2 lautet somit:

H2: Der Einsatz von OpenProposal führt zu konstruktiven Anwenderbeiträgen.

OpenProposal wurde so konstruiert, dass der Anwender seine Anwenderbeiträge auch während der alltäglichen Nutzung seiner Software und in Abwesenheit des Moderators erstellen sowie über eine Internet- bzw. Netzwerkverbindung an den Moderator versenden kann. Der Prozess zur Erstellung von Anwenderbeiträgen wurde auf ein Minimum an notwendigen Arbeitsschritten reduziert, um dem Anwender die Feedbackabgabe zu erleichtern. Hypothese H3 lautet somit:

H3: OpenProposal motiviert Anwender dazu, Beiträge im Betriebsalltag zu erstellen.

Die Akzeptanz von OpenProposal durch den Moderator soll durch die Reduzierung des Aufwandes zur Vor- und Nachbereitung der Anwenderbeteiligung sichergestellt werden. Der Einsatz von OpenProposal lässt erwarten, dass der Moderator einen geringen Aufwand für die Erhebung von Anwenderbeiträgen benötigt. Hypothese H4 lautet somit:

H4: OpenProposal reduziert den Aufwand der Anwenderbeteiligung.

Eine Erhöhung der Anzahl der Anwenderbeiträge, wie sie unter H1 erläutert wurden, kann jedoch nur dann hilfreich sein, wenn sich gleichzeitig auch die Qualität der Anwenderbeiträge verbessert oder zumindest nicht verschlechtert. Es wird erwartet, dass die mit OpenProposal erstellten Anwenderbeiträge verständlich sind. Hypothese H5 lautet somit:

H5: Die Verständlichkeit der mit OpenProposal erstellten Anwenderbeiträge ist hoch.

Zur Überprüfung der fünf Hypothesen H1 – H5 in den Fallstudien wird als unabhängige Variable UV1 die Wahl der Methode zur Anwenderbeteiligung festgelegt. Dies inkludiert Methoden sowohl zur asynchronen als auch zur synchronen Anwenderbeteiligung, um die Ergebnisse der Literaturanalyse aus Kapitel 3 und 4 zu prüfen und OpenProposal mit den existierenden Methoden zu vergleichen:

UV1: Wahl der Methode zur Anwenderbeteiligung

Die Wahl der Methode zur Anwenderbeteiligung wirkt sich nach den Bewertungskriterien aus Kapitel 2.9 auf die Effizienz der Entwicklung, der Qualität der Anwenderbeiträge und auf die Belastung der Anwender aus. Für die Prüfung der Hypothesen H1 - H5 sind daher abhängige Variablen zu definieren, die sich auf diese Bewertungskriterien beziehen.

Für Hypothese H1 ist die Anzahl der erstellen Anwenderbeiträge relevant. Hieraus folgt die abhängige Variable AV1:

AV1: Anzahl der erstellten Anwenderbeiträge

Bei Hypothese H2 ist Umfang und Konstruktivität der Anwenderbeiträge anzuwenden. Dies führt zu den abhängigen Variablen AV2 und AV3:

AV2: Umfang der erstellten Anwenderbeiträge
AV3: Konstruktivität der erstellten Anwenderbeiträge

Hypothese H3 lässt sich anhand der Effizienz und Komplexität der Erstellung von Anwenderbeiträgen sowie der Anzahl der erstellten Anwenderbeiträge (vgl. AV1) und der Zufriedenheit der Anwender mit der Methode zur Anwenderbeteiligung bewerten. Die abhängigen Variablen AV4, AV5 und AV6 lauten somit:

AV4: Aufwand des Anwenders zur Erstellung von Anwenderbeiträgen
AV5: Komplexität der Erstellung von Anwenderbeiträgen
AV6: Zufriedenheit der Anwender mit der Methode zur Anwenderbeteiligung

Für Hypothese H4 ist die Effizienz der Vor- und Nachbereitung der Anwenderbeteiligung sowie der Interaktion mit den Anwendern zu messen. Da sich die Effizienz durch den notwendigen zeitlichen und materiellen Aufwand bemisst, lauten die abhängigen Variablen AV7, AV8 und AV9:

AV7: Aufwand des Moderators für die Vorbereitung zur Erfassung von Anwenderbeiträgen
AV8: Aufwand des Moderators für die Nachbereitung der erstellten Anwenderbeiträge
AV9: Aufwand des Moderators für die Interaktion mit den Anwendern

Um Hypothese H5 prüfen zu können, ist die Verständlichkeit der Anwenderbeiträge von Bedeutung. Die abhängige Variable AV10 lautet somit:

AV10: Verständlichkeit der erstellten Anwenderbeiträge

Tabelle 9.1 fasst die Auswahl der Metriken zusammen und beschreibt die oben aufgeführten Zusammenhänge zwischen Hypothesen und abhängigen Variablen.

Hypothese	Abhängige Variable
H1: Mit OpenProposal werden im Vergleich zu anderen Methoden mehr Anwenderbeiträge erfasst.	- AV1: Anzahl der erstellten Anwenderbeiträge
H2: Der Einsatz von OpenProposal führt zu konstruktiven Anwenderbeiträgen.	- AV2: Umfang der erstellten Anwenderbeiträge - AV3: Konstruktivität der erstellten Anwenderbeiträge
H3: OpenProposal motiviert Anwender Anwenderbeiträge im Betriebsalltag zu erstellen.	- AV1: Anzahl der erstellten Anwenderbeiträge - AV4: Aufwand des Anwenders zur Erstellung von Anwenderbeiträgen - AV5: Komplexität der Erstellung von Anwenderbeiträgen - AV6: Zufriedenheit der Anwender mit der Methode zur Anwenderbeteiligung
H4: OpenProposal reduziert den Aufwand der Anwenderbeteiligung.	- AV7: Aufwand des Moderators für die Vorbereitung zur Erfassung von Anwenderbeiträgen - AV8: Aufwand des Moderators für die Nachbereitung der erstellten Anwenderbeiträge - AV9: Aufwand des Moderators für die Interaktion mit den Anwendern
H5: Die Verständlichkeit der mit OpenProposal erstellten Anwenderbeiträge ist hoch.	- AV10: Verständlichkeit der erstellten Anwenderbeiträge

Tabelle 9.1: Bewertungskriterien für Methoden zur Anwenderbeteiligung

9.3.3 Gegenstand der Untersuchungen

Die Untersuchungen zu OpenProposal beziehen sich auf die Prozesse der Erfassung und Bearbeitung von Anwenderbeiträgen von Anwendern in Software-Projekten. Involviert sind Anwender, Entscheider, Moderatoren und Entwickler. Eine Anwenderbeteiligung kann zu unterschiedlichen Zeitpunkten erfolgen: Zu Beginn eines Software-Projektes werden Anforderungen erhoben; im Rahmen von Be-

nutzertests können Prototypen von der ersten bis zur finalen Version getestet werden; bei der Einführung von Software-Produkten in Testinstallationen wird die frühe Betriebsphase untersucht; in der produktiven Umgebung können während des Betriebs weitere Untersuchungen stattfinden. Als Software-Produkte sollen hier nur Anwendungen verstanden werden, die an einem Computer bedient und folglich mit dem OpenProposal Annotationssystem annotiert werden können. Hierfür haben die Anwender die Möglichkeit, die zu testende Software an einem Computer zu benutzen und währenddessen Anwenderbeiträge zu erstellen.

Eine Einschränkung auf eine bestimmte Organisationseinheit ist nicht notwendig, da die Zusammensetzung eines Teams in einem Software-Projekt zwar auf Rollen abstrahiert aber im Allgemeinen nicht in Organisationseinheiten unterteilt werden kann. Es wird auch kein spezifisches Vorgehensmodell vorausgesetzt, da die etablierten Vorgehensmodelle des Software Engineering wie z.B. V-Modell XT und RUP die Anwenderbeteiligung lediglich als optionales Mittel erwähnen und weder eine Anleitung noch eine Struktur für den Ablauf einer Anwenderbeteiligung vorgeben.

9.3.4 Auswahl der Fallstudien

Um die aufgestellten Hypothesen zu prüfen und Antworten auf die Forschungsfragen zu finden, sollen die Fallstudien die Möglichkeit zur Erprobung von OpenProposal im Rahmen von realen Software-Projekten bieten, so dass aus den beobachteten Effekten aufschlussreiche Erkenntnisse gewonnen werden können. Voraussetzung für die Fallstudien ist, dass

- der Zugang zum Forschungsfeld frei zugänglich und zur Auswertung freigegeben ist (vgl. [Kitchenham und Pickard, 1995]),
- eine reibungslose - sowohl technische als auch organisatorische - Integration von OpenProposal in die Arbeitsprozesse sichergestellt werden kann,
- die Erstellung von Anwenderbeiträgen zu Software-Produkten sowohl synchron, z.B. in Form von Interviews und Workshops, als auch zeitlich und bzw. oder räumlich asynchron beispielsweise im Betriebsalltag erfolgen können, und
- die Untersuchungen im Rahmen eines realen Software-Projektes stattfinden können.

Die Forscher[27] nehmen als neutrale Beobachter an der Fallstudie teil und bemühen sich bei ihren Beobachtungen, Befragungen und Auswertungen um einen geringstmöglichen Einfluss auf den Ablauf der Studie.

Im Laufe der Entwicklung von OpenProposal konnten zwei Fallstudien durchgeführt werden, die den oben genannten Auswahlkriterien entsprechen. Die beiden Fallstudien wurden unabhängig voneinander und zeitlich versetzt durchgeführt. Im Sinne einer fallübergreifenden Fallstudienanalyse sollen dabei die Ergebnisse der ersten Fallstudie in der zweiten Fallstudie repliziert werden.

Die Möglichkeit zur Durchführung der beiden Fallstudien ergab sich auf Anfragen des Unternehmens TRUMPF und aus dem Forschungsprojekt WAVES. In den Fallstudien wurde jeweils eine Phase eines

[27] Neben dem Autor dieser Arbeit wirkten Studenten unterstützend mit.

Software-Projektes betrachtet und nicht ein ganzes Software-Projekt von Anfang bis Ende verfolgt. Die Motivation zur Erprobung von OpenProposal ging von den Moderatoren aus. Bei TRUMPF handelte es sich bei der Moderatorin um die Usability Expertin des Unternehmens und die Organisatorin von Usability Workshops. In WAVES war die Moderatorin eine Mitarbeiterin des Evaluationspartners des Forschungsprojekts. Beide waren in den Software-Projekten für die Erfassung von Anwenderbeiträgen zuständig.

Im Vorfeld der Fallstudie „TRUMPF“ (vgl. Kapitel 9.4 auf Seite 171) hatte das Unternehmen TRUMPF Maschinenwerkzeuge aus Ditzingen mit dem OpenProposal Team Kontakt aufgenommen und um einen Einsatz von OpenProposal bei der Durchführung von Usability Workshops gebeten. Die Studie begann im September 2007 und wurde im Oktober desselben Jahres abgeschlossen. Ein Jahr später wurde OpenProposal auf Wunsch des Projektteams im Forschungsprojekt WAVES in der zweiten Fallstudie „WAVES“ (vgl. Kapitel 9.5 auf Seite 188) erprobt. Die Studie wurde von Juli bis Oktober 2008 durchgeführt. Im Rahmen des Forschungsprojektes WAVES unterstützte OpenProposal die Evaluation der im Forschungsprojekt implementierten Anwendungen. Im Rahmen von Benutzertests vor Ort und im laufenden Betrieb wurde OpenProposal zur Erhebung von Verbesserungsvorschlägen und Fehlermeldungen zu einer integrierten Suchmaske sowie einem Rich-Wiki Client verwendet.

In beiden Fallstudien wurde OpenProposal somit zur Erfassung von Anwenderbeiträgen zu einem neuen Software-Produkt eingesetzt. Die getesteten Software-Produkte befanden sich zum Zeitpunkt des Beginns der Fallstudie in einem frühen Entwicklungsstadium. Der freie Zugang zum Forschungsfeld und zur Auswertung wurde ermöglicht. Hierfür wurde OpenProposal in die bestehenden Prozesse und die Entwicklungsumgebung integriert. Während der Durchführung der Fallstudien bestand die Aufgabe der OpenProposal Forscher darin, Befragungen durchzuführen und die an der Fallstudie beteiligten Anwender zu beobachten. Allerdings waren die Untersuchungen in der ersten Studie nur auf einen zwei-tägigen Usability Workshop beschränkt, so dass OpenProposal nicht im realen Betriebsalltag der Software-Nutzung eingesetzt werden konnte. Dies führt dazu, dass die Hypothese H3 in der Fallstudie TRUMPF lediglich unter Laborbedingungen untersucht werden konnte, während alle anderen Hypothesen in der Fallstudie TRUMPF untersucht werden konnten. Die Fallstudie WAVES erfüllt alle Auswahlkriterien und erlaubt

9.3.5 Vorgehensweise der Datenerhebung und Auswertung

Für jede Einzelfallstudie werden in den Kapiteln 9.4 (vgl. Seite 171ff) und 9.5 (vgl. Seite 188ff) der Hintergrund der Studie aufbereitet, das Design der Studie vorgestellt und anschließend die Ergebnisse der Datenerhebung präsentiert. In einer fallstudienübergreifenden Betrachtung werden die Kernergebnisse aus den Fallstudien in Kapitel 9.6 (vgl. S. 214ff) zusammengefasst und die Hypothesen H1 - H5 diskutiert.

Die Vorgehensweise bei der Datenerhebung orientierte sich in beiden Studien an derselben Struktur:

1. Anwender testeten im Rahmen eines real existierenden Software-Projektes einen Software-Prototyp, entweder im Rahmen eines Usability Workshops oder einer Reihe von persönlichen Einzelinterviews. In der Fallstudie TRUMPF blieben die Forscher im Hintergrund und notierten sich ihre Beobachtungen schriftlich. Bei der Fallstudie WAVES nahmen die Forscher auf Wunsch der Moderatorin hin nicht am Test teil. Der Test wurde auf Video aufgezeichnet und im Nachhinein ausgewertet. Jeder Test durchlief das gleiche Muster:

1.1. Die Anwender erhielten eine kurze Einführung über die zu testende Software, den Hintergrund der Studie sowie das Werkzeug OpenProposal.

1.2. Die Anwender führten verschiedene Aufgaben durch, anhand derer die Software getestet werden sollte, und wurden gebeten Anwenderbeiträge mit dem OpenProposal Annotationssystem abzugeben. Zum Vergleich wurden auch alternative Methoden eingesetzt.

1.3. Anschließend werden sowohl die zu untersuchende Software als auch die Nützlichkeit von OpenProposal von den Anwendern diskutiert. Außerdem werden die erstellten Vorschläge bezüglich ihrer Verständlichkeit sowie hinsichtlich eventueller Auffälligkeiten untersucht.

1.4. Am Ende jedes Anwendertests werden die Anwender gebeten, einen Fragebogen auszufüllen.

2. Die Anwender wurden zudem darum gebeten, den Prototyp im Rahmen ihrer täglichen Arbeit zu testen und wahlweise OpenProposal oder andere Methoden zu verwenden um ihre Anwenderbeiträge abzugeben. In der Fallstudie TRUMPF war dies nicht möglich. In der Fallstudie WAVES konnte eine dreimonatige Testphase durchgeführt werden.

3. Nach Abschluss des Software-Projekts wurden die Moderatoren, Entscheider und Entwickler in einem leitfadengestützten Interview persönlich nach ihrer Einschätzung von OpenProposal befragt. Jedes Interview wurde nachträglich schriftlich zusammengefasst und die Zusammenfassung von der befragten Person auf ihre Richtigkeit hin überprüft.

4. Die Auswertung der Fallstudien umfasste die Notizen und Aufzeichnungen zu den Beobachtungen und Befragungen, die erstellten Verbesserungsvorschläge und die ausgefüllten Fragebögen der Anwender.

5. Die Ergebnisse der Auswertungen wurden für jede Fallstudie gesondert in einem Fallstudienbericht zusammengefasst und den Teilnehmern der Fallstudien zur Validierung zur Verfügung gestellt.

6. Zur Interpretation der Ergebnisse wurden die eingangs aufgestellten Forschungsfragen und Hypothesen aufgegriffen. Mit den Auswertungen erfolgte die Interpretation der Ergebnisse zur Verifikation bzw. Falsifikation der Hypothesen. Darauf aufbauend wurden Antworten auf die Forschungsfragen formuliert. Die Interpretationen wurden mit zwei anderen Forschern am FZI Forschungszentrum Informatik mit Expertise im Software Engineering diskutiert und in Publikationen und Vorträgen sowohl wissenschaftlichen als auch praxisnahen Foren präsentiert.

Tabelle 9.2 fasst die wesentlichen Merkmale der beiden Fallstudien zusammen. Die Tests wurden von dem jeweiligen Moderator vorbereitet und ausgewertet. Jede teilnehmende Person war nur an einer der beiden Fallstudien beteiligt. Keiner der Anwender hatte OpenProposal zuvor verwendet. In beiden Studien hatten die Entwickler lediglich über Telefon, Email und ein Issue-Trackersystem Kontakt mit Anwendern. Die Entwickler von OpenProposal waren als reine Beobachter vor Ort und weder an den Software-Projekten noch an der Organisation der Befragungen beteiligt. Für die Verwaltung von

Aufgaben und Anwenderbeiträge wurde in beiden Fällen sowohl von den Teammitgliedern des Software-Projektes als auch von den Anwendern selbst das webbasierte Issue-Tracker-System JIRA verwendet. Mit OpenProposal konnten Anwenderbeiträge zur weiteren Bearbeitung direkt an JIRA gesendet werden.

Fallstudie	# Unternehmen, # Mitarbeiter	# Teilnehmer	Hintergrund Anwender	Softwaretyp	Umfang
„TRUMPF“	1 mittelständisches Unternehmen, 5.000 - 10.000 Mitarbeiter	11 Anwender 1 Moderator 1 Entscheider 1 Entwickler	Training, Vertrieb, Techn. Doku, Support	3D-Konstruktion	1x 2 Tage Usability Workshop
„WAVES“	4 kleine Unternehmen, jw. 1 - 500 Mitarbeiter	16 Anwender 1 Moderator 4 Entscheider 4 Entwickler	Software-Entwickler	Wiki, Suchmaschine	4x 1 Tag Interviews und 3 Monate im Betriebsalltag

Tabelle 9.2: Übersicht über die Fallstudien

9.3.6 Verknüpfungslogik der Daten mit den Hypothesen

Die in den Fallstudien erfassten Daten lagen in Form von Beobachtungsprotokollen, Fragebögen, Befragungsprotokollen und den erstellten Anwenderbeiträgen vor. Die Datenerhebung mithilfe von Fragebögen sowie Befragungen diente der Untersuchung der Variablen AV1 - AV10. Anhand von Beobachtungsprotokollen konnten Daten zu den Variablen AV4 - AV10 ermittelt werden. Die Analyse der erstellten Anwenderbeiträge lieferte Daten zu den Variablen AV1 - AV4, AV8 und AV10.

AV1 ergibt sich durch das Zählen aller erstellten Verbesserungsvorschläge. Der Umfang der erstellten Verbesserungsvorschläge in AV2 wird durch die Menge an Text und ergänzenden Informationen ermittelt. Hierfür wird die Anzahl der abgegebenen Wörter und zusätzlichen Informationen wie z.B. Zeichnungen, Protokolle, etc. bestimmt. Die Konstruktivität der erstellten Anwenderbeiträge bei AV3 lässt sich quantitativ nur schwierig messen, da kein Maß für den Grad der Konkretheit eines Textes existiert. Stattdessen wird im Rahmen von Befragungen der Moderatoren, Entscheider und Entwickler nach der subjektiven Wahrnehmung der Konstruktivität gefragt. AV5, AV7, AV8 und AV9 drücken sich im zeitlichen Aufwand aus und werden in Befragungen, im Fragebogen und aus den Beobachtungen ermittelt. Die Zufriedenheit der Anwender mit der Wahl der Methode bei AV6 wird im Rahmen der Beobachtungen sowie in Form einer Schulnote durch einen Fragebogen erfasst. Die Verständlichkeit der Anwenderbeiträge in AV10 ergibt sich aus Befragungen der Entscheider und Entwickler.

Diese Variablen bilden die Grundlage für die Untersuchung der Hypothesen H1 – H5. Zur Untersuchung der Hypothese H1 dient die Variable AV1. Es wird angenommen, dass H1 bestätigt werden kann, wenn in den Fallstudien die Variable AV1 für OpenProposal einen höheren Wert als für die anderen Methoden beträgt.

Die Hypothese H2 findet Unterstützung, wenn mit OpenProposal im Vergleich zu anderen Methoden konstruktivere Verbesserungsvorschläge erstellt werden. Außerdem sollten mit OpenProposal erstellte Beiträge im Verhältnis keine bzw. so wenig Fehlermeldungen und so wenig allgemeine Fragen wie möglich aufwerfen. Im Vergleich zu anderen Werkzeugen sollte stattdessen ein höherer Anteil an konstruktiven neuen Ideen feststellbar sein. Dies wird durch die Variable AV3 ausgedrückt.

Die Hypothese H3 kann bestätigt werden, wenn beim Einsatz von OpenProposal im Betriebsalltag die Variablen AV1 und AV6 hoch sowie AV4 und AV5 niedrig sind, d.h. es werden Anwenderbeiträge im Betriebsalltag mit niedrigem Aufwand und geringer Komplexität erstellt und der Anwender ist zufrieden.

Für die Bestätigung von Hypothese H4 müssen die Variablen AV7, AV8 und AV9 beim Einsatz von OpenProposal in der Summe geringer sein als bei den anderen Methoden.

Um Hypothese H5 bestätigen zu können sollte die Variablen AV10 für die OpenProposal Anwenderbeiträge höher als für anders erstellte Beiträge sein.

Variablen	Adressierte Hypothesen	Datenquellen
AV1	H1	Anwenderbeiträge
AV2, AV3	H2	Anwenderbeiträge, Befragungen, Beobachtungen
AV1, AV4, AV5, AV6	H3	Fragebogen, Befragungen, Beobachtungen
AV7, AV8, AV9	H4	Befragungen, Beobachtungen
AV10	H5	Befragungen

Tabelle 9.3: Verknüpfung der Variablen, Hypothesen und Datenquellen in den Fallstudien

Tabelle 9.3 fasst die Verknüpfungen zwischen den abhängigen Variablen, den Hypothesen, den Datenquellen und deren Bedeutung für die Analyse der Fallstudien zusammen.

9.3.7 Kriterien zur Interpretation der Ergebnisse

Die Interpretation der Ergebnisse stützte sich auf die Auswertung der erhobenen Daten aus den beiden Fallstudien. Mithilfe der Technik des Mustervergleichs (Yin 2003 S. 116ff) wurden anhand der abhängigen Variablen die zu erwartenden Ergebnisse als Muster bestimmt. Anhand dieser Kriterien erfolgte die Verifikation bzw. Falsifikation der Hypothesen.

Im Idealfall sollten alle fünf Hypothesen H1 - H5 bestätigt werden. Dies würde bedeuten, dass OpenProposal bei allen Beteiligten auf Akzeptanz gestoßen ist. In diesem Fall würden die Gründe in der im Vergleich zu anderen Methodenliegen höheren Effizienz und Effektivität der Anwenderbeteiligung.

Können eine oder mehrere der Hypothesen nicht bestätigt werden, sollten für jede Falsifikation einer Hypothese deren Ursachen diskutiert und daraus Rückschlüsse auf die Korrektheit der Anforderungsanalyse und die Konzipierung von OpenProposal gezogen werden. Sollten alle Hypothesen falsifiziert

werden, ist davon auszugehen, dass OpenProposal nicht zu einer Verbesserung der Anwenderbeteiligung führt.

9.4 Die Fallstudie „TRUMPF“[28]

Die Fallstudie „TRUMPF“ stellt die erste Fallstudie der Evaluation von OpenProposal Methode dar. Im Folgenden werden zu Beginn der Unternehmenskontext vorgestellt, dann das Design der Studie präsentiert und abschließend die Ergebnisse zusammengefasst und zur Diskussion gestellt.

9.4.1 Hintergrund zur Fallstudie „TRUMPF“

Die Firma TRUMPF ist ein mittelständisches Technologieunternehmen, dessen Schwerpunkte auf Fertigungstechnik wie Werkzeugmaschinen, Elektrowerkzeuge, Lasertechnik und Elektronik, sowie auf Medizintechnik liegen. Werkzeugmaschinen und Elektrowerkzeuge für die Blechbearbeitung zum Stanzen, Biegen, Schneiden und Schweißen sind dabei für den größten Umsatz bei TRUMPF verantwortlich. Mit einem Umsatz von 1.937,9 Millionen € und 7.258 Mitarbeiter (Stand: 01. Oktober 2007) gehört das global agierende Familienunternehmen aus Ditzingen zu den Weltmarktführern auf diesem Gebiet.

Die von TRUMPF hergestellten Maschinen werden seit 1979 über die mitgelieferte Software TruTops konfiguriert und gesteuert. Die Software wird von der firmeneigenen Software-Entwicklungsabteilung in Ditzingen entwickelt. Für die unterschiedlichen Maschinen und Einsatzzwecke stellt die Software-Entwicklungsabteilung spezielle Software-Module (u.a. TruTops Laser, TruTops CAD, TruTops Tube, etc.) zur Verfügung, mit der Einstellungen an den Maschinen vorgenommen und die Produktion programmiert und verwaltet werden können. Abbildung 9.2 zeigt beispielhaft ein Bildschirmfoto einer solchen Anwendung. Die zu produzierende Werkstücke werden grafisch abgebildet und an die Produktionsplanung übergeben.

Seit Juli 2006 steht bei TRUMPF die Verbesserung der Bedienfreundlichkeit der Software im Vordergrund der Entwicklungen. Kernelement dieser Bemühungen ist die Durchführung von Usability Workshops, in denen die Bedienbarkeit getestet und Verbesserungsvorschläge erhoben werden sollen.

Diese Usability Workshops finden, wie in Abbildung 9.3 dargestellt, stets kurz vor dem Ende der Entwicklung („Code Close“) statt. In einem iterativen Software-Entwicklungsprozess wird das aktuelle Release der Software-Module von der Software-Entwicklungsabteilung entwickelt. Erweisen sich die Funktionstests als erfolgreich, wird das aktuelle Release für einen Usability Workshop freigegeben. Die im Usability Workshop generierten Überarbeitungsvorschläge werden im Anschluss implementiert. Das Release wird schließlich für abgeschlossen erklärt. Es folgen interne Tests bei TRUMPF und bei ausgesuchten Kunden. Verlaufen die Tests erfolgreich, erfolgt die Freigabe für die Serienausgabe.

[28] Ein großer Dank gebührt an dieser Stelle den Herren Jan Baumann, David Meder und Ronald Walenda, die mit ihren Bachelor- und Diplomarbeiten maßgeblich zur erfolgreichen Durchführung der Fallstudie beigetragen haben. Außerdem ist dem Unternehmen TRUMPF für die Ermöglichung der Studie herzlichst zu danken.

Die Workshops werden in den Schulungsräumen von TRUMPF durchgeführt und dauern einen bis drei Tage. Üblicherweise nehmen fünf bis 15 Personen an den Workshops statt. In den Workshops wird mit einer Einführung in das zu testende Programm begonnen und die Software anschließend anhand von Aufgaben bedient und getestet. Während der Bearbeitung der Aufgaben sollen die Teilnehmer alle Fehler, Kommentare und Verbesserungsvorschläge notieren und in der Abschlussrunde zur Diskussion stellen.

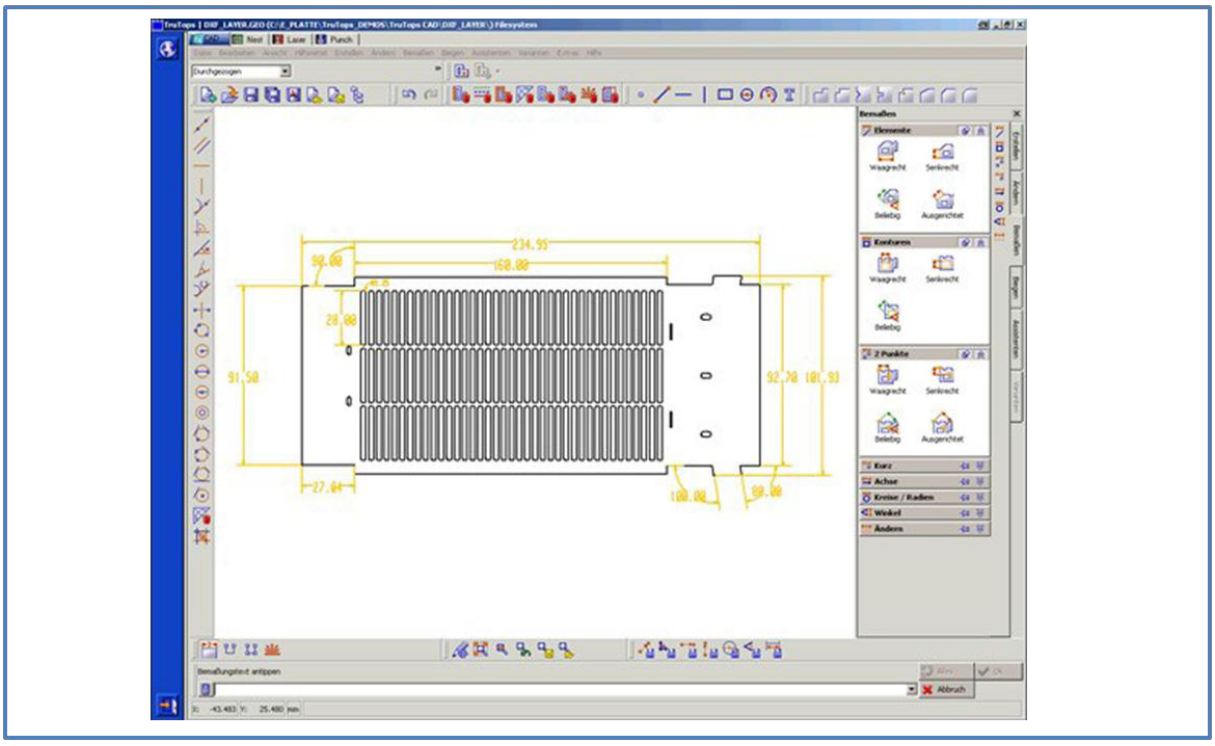

Abbildung 9.2: Bildschirmfoto einer TRUMPF Software

Ein Problem für das Entwicklungsteam stellte dabei bisher die aufwändige papierbasierte Erfassung der Anwenderbeiträge und die Dokumentation der Ergebnisse der Workshops dar. Bis zum Zeitpunkt der OpenProposal-Studie notierten die Tester ihre Beiträge während des Workshops handschriftlich auf Notizzettel. Dies stellte das Entwicklerteam vor das Problem, dass diese Notizen sehr aufwändig und mit vielen Fehlern behaftet in eine elektronisch geführte Tabelle übertragen werden mussten. Außerdem konnte die Bedeutung der Notizen häufig nicht mehr nachvollzogen werden.

Daher suchte das Team nach anderen Möglichkeiten zur Erfassung von Anwenderbeiträgen. Beispielsweise war den Testpersonen die Möglichkeit gegeben worden, elektronische Textdokumente, Bildschirmfotoprogramme oder Issue-Tracker zur Erfassung ihrer Vorschläge zu verwenden. Dieser Ansatz wurde von den Testern jedoch nicht genutzt, da diese den ständigen Wechsel zwischen der zu testenden Anwendung und dem System zur Erstellung von Beiträgen als zu umständlich und für den Testablauf als störend empfanden. Die Möglichkeit, Bildschirmfotos auszudrucken und mit einem Stift zu annotieren, wurde ebenfalls nicht akzeptiert. Daher erwägte TRUMPF im Juli 2007 den Einsatz von OpenProposal zur Unterstützung der Usability Workshops. Die Verantwortlichen waren über eine Publikation von Rashid et al. [2006b] auf OpenProposal aufmerksam geworden und nahmen daher Kontakt mit den OpenProposal Forschern auf.

Der Usability Workshop im eigentlichen Sinn stellt eine synchrone Methode dar. Allerdings war TRUMPF der Auffassung, dass die Anwender zwar alle gleichzeitig in einem Raum sitzen, aber asynchron testen und ihre Beiträge asynchron erstellen. Die Usability Workshops werden mit mehreren Testpersonen gleichzeitig durchgeführt, um von unterschiedlichen Testpersonen Feedback zu erhalten und in diesem Rahmen gemeinsame Diskussionen zum Ende des Workshops führen zu können. Allerdings stehen nicht genügend Ressourcen zur Verfügung, um jeden Anwender von einem Moderator betreuen zu können, sodass lediglich eine einzige Moderatorin für die Betreuung der Anwender anwesend ist. Daher ist in diesem Fall von einer Mischung asynchroner und synchroner Methoden auszugehen. OpenProposal sollte dabei zur Lösung der Probleme mit der asynchronen Erfassung der Anwenderbeiträge beitragen.

In einem ersten Treffen wurde OpenProposal der Moderatorin, dem Entscheider und den Entwicklern der Software-Entwicklungsabteilung von TRUMPF präsentiert und die Gestaltung der Studie besprochen. Die Moderatorin war eine Mitarbeiterin der Abteilung und für die Verbesserung der Bedienbarkeit der TRUMPF Software verantwortlich. Als promovierte Psychologin war sie auf die Gestaltung und Verbesserung von Benutzerschnittstellen von Software- und Hardware-Systemen spezialisiert. Beim Entscheider handelte es sich um den Abteilungsleiter der Software-Entwicklungsabteilung. Die Entwickler stammten ausschließlich aus der Software-Entwicklungsabteilung.

Als Vergleichssystem zum OpenProposal Annotationssystem sollte der Issue-Tracker JIRA herangezogen werden. Zur Abgabe von Anwenderbeiträgen diente eine webbasierte Eingabemaske, die neben der Abgabe von textlich formulierten Vorschlägen auch das Anhängen von Bildschirmfotos bzw. Abbildungen ermöglichte. JIRA war ein halbes Jahr zuvor für das interne Aufgabenmanagement bei der Software-Entwicklungsabteilung von TRUMPF eingeführt und seither von den Mitarbeitern im Rahmen ihrer täglichen Arbeiten verwendet worden. JIRA wurde als Alternative in Erwägung gezogen, da es bei den Mitarbeitern der Software-Entwicklungsabteilung bereits gut akzeptiert wurde und auch für die Erstellung von Anwenderbeiträgen genutzt werden kann.

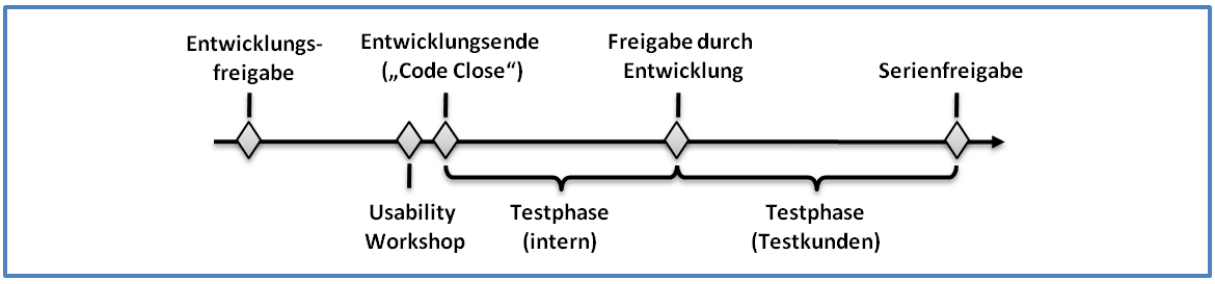

Abbildung 9.3: TRUMPF Software-Entwicklungsprozess

9.4.2 Gestaltung der Fallstudie „TRUMPF“

Die Planung und Durchführung der Fallstudie wurde in enger Abstimmung mit den Mitarbeitern von TRUMPF vorgenommen. Die Gestaltung der Fallstudie umfasste dabei die Beschreibung der Ziele, die Auswahl der Teilnehmer, die Installation des technischen Aufbaus und die Planung des zeitlichen Ablaufs der Fallstudie.

9.4.2.1 Ziele der Fallstudie „TRUMPF“

Auf Wunsch von TRUMPF sollten im Rahmen eines Usability Workshops die Vor- und Nachteile von OpenProposal untersucht und das OpenProposal Annotationssystem einem Vergleich zu JIRA unterzogen werden. In diesem Zusammenhang sollte eine neue Software von TRUMPF getestet werden, indem die Anwender Aufgaben bearbeiten sollten, die an typischen Aufgabenstellungen ihrer Kunden angelehnt sind. Sobald ein Fehler oder eine Verbesserungsmöglichkeit entdeckt würde, sollte dieser entweder mit OpenProposal oder JIRA gemeldet werden. Bei sehr schweren Programmfehlern sollten die Anwender diese für alle Tester sichtbar auf einem Whiteboard platzieren, um so zu verhindern, dass derselbe Fehler ein zweites Mal angemerkt würde. Am Ende jedes Workshoptages wurde eine gemeinsame Diskussion zum Ablauf und die Einschätzung des vergangenen Tages anberaumt.

In den Untersuchungen zu OpenProposal sollte Bezug nehmend auf die Hypothesen H1 - H5 der Frage nachgegangen werden, ob Anwender dazu motiviert werden können, ihre Beiträge mittels OpenProposal elektronisch abzugeben, und ob dies dazu beitragen kann, die Aufwände für Usability Workshops bei TRUMPF zu reduzieren und dabei auch deren Ergebnisse zu verbessern. Hypothese H3 konnte dabei nur bedingt untersucht werden, da sich die Studie auf Wunsch der TRUMPF nur auf Usability Workshops konzentrierte und somit der Einsatz von OpenProposal im Betriebsalltag nur unter Laborbedingungen untersucht werden konnte. Da die Anwender bei der Erstellung der Anwenderbeiträge allein auf sich gestellt sind und die gestellten Aufgaben aus dem Betriebsalltag entnommen sind, ist von einer künstlichen Nachbildung des Betriebsalltags auszugehen.

Für die unabhängige Variable UV1 wurde zwischen „UV1-1 OpenProposal“ und „UV1-2 JIRA“ variiert.

9.4.2.2 Teilnehmer der Fallstudie „TRUMPF“

Die Teilnehmer der Fallstudie setzten sich aus elf Anwendern (vier weiblich, sieben männlich), einer Moderatorin, einem Entscheider und einem Entwickler zusammen. Der Altersdurchschnitt der Anwender lag bei 29,75 (MW: 29.75; MIN: 27, MAX: 47; STABW: 6,84). Sie stammten aus unterschiedlichen Abteilungen von TRUMPF und wiesen unterschiedliche Erfahrungen im Umgang mit der Formulierung von Anforderungen auf; fünf schätzen sich als erfahren und sechs als unerfahren ein.

Keine der Testpersonen hatte die Testsoftware oder OpenProposal zuvor verwendet. Da JIRA bereits vor der Studie für das Melden von Problemen in Form eines Helpdesk und für die Verwaltung von Aufgaben verwendet worden war, wurde davon ausgegangen, dass alle Testpersonen mit JIRA vertraut waren. Außerdem waren sie an der Entwicklung, Schulung und dem Vertrieb der TRUMPF Produkte beteiligt, so dass sie auch mit der Produktpalette der TRUMPF Software sowie mit der Konfiguration der Maschinen des Unternehmens vertraut sein mussten. Anhand der Fragebögen konnte die Vertrautheit der Testpersonen mit JIRA und der Software von TRUMPF bestätigt werden.

Die Testpersonen wurden zu Beginn des Workshops zufällig in zwei nahezu gleichgroße Gruppen, Gruppe A und B, eingeteilt. Am ersten Tag sollte Gruppe A OpenProposal nutzen und Gruppe B JIRA. Am zweiten Tag tauschten die Gruppen. Am Ende jedes Tages erhielten die Testpersonen einen Fragebogen zu dem Werkzeug, das sie an diesem Tag verwendet hatten. Durch diese Aufteilung sollte sichergestellt werden, dass jede Testperson beide Werkzeuge bewerten konnte und mithilfe der Gesamtbetrachtung beider Gruppen die Bewertung nicht durch die Reihenfolge des Einsatzes beeinflusst wurde.

9.4.2.3 Technischer Aufbau der Fallstudie „TRUMPF“

Die Testpersonen saßen gemeinsam in einem großen Schulungsraum bei TRUMPF. Jede Person hatte einen eigenen Arbeitsplatz mit einem Computer mit Netzwerkverbindung zur Verfügung. Auf jedem Computer konnten OpenProposal und JIRA bei Bedarf über eine Verknüpfung aufgerufen werden. Als eine weitere Möglichkeit zur Feedbackabgabe befand sich ein Whiteboard im Raum, auf dem Hinweise und Fehler auf Kärtchen für alle sichtbar angeheftet werden konnten.

OpenProposal lag zum Zeitpunkt der Fallstudie in der Version 2.0 vor. Bis auf die automatische Generierung des Titels und der Kurzbeschreibung wurden alle Konzepte zu OpenProposal implementiert. Die Anwender mussten bei ihren Vorschlägen daher zusätzlich einen Titel und eine allgemeine Kurzbeschreibung ihres Anwenderbeitrags eingeben.

Mehrere Tage vor dem Usability Workshop wurde OpenProposal in die Entwicklungsumgebung von TRUMPF integriert. Hierzu musste eine Schnittstelle zu JIRA in OpenProposal implementiert werden, so dass die mit OpenProposal erstellten Anwenderbeiträge beim Absenden automatisch als Issue in JIRA eingetragen wurden. Darüber hinaus wurden Regeln definiert, nach denen die Vorschläge in JIRA abhängig vom gewählten Annotationswerkzeug und dem annotierten Modul der Testsoftware automatisch einem Projekt und einer Kategorie zugeordnet wurden.

JIRA stand als webbasierte Anwendung zur Verfügung und konnte über einen Internetbrowser aufgerufen werden. Zur Erstellung von Anwenderbeiträgen meldete sich der Anwender über die Anmeldemaske von JIRA an und drückte den Button „Neuer Vorgang“. Hier wurden zuerst das zugehörige Projekt und die Kategorie ausgewählt und anschließend ein Titel sowie eine Beschreibung eingegeben. Optional konnte der Anwender über einen zusätzlichen Interaktionsdialog in JIRA ein Bildschirmfoto erstellen und dieses in ein dafür vorgesehenes Textfeld einfügen.

9.4.2.4 Zeitlicher Ablauf der Fallstudie „TRUMPF“

Die Vorbereitungen zum Workshop begannen im Juli 2007. Zu Beginn wurde die technische Integration von OpenProposal mit dem Issue-Tracker von TRUMPF durchgeführt und die technische Funktionalität von OpenProposal sichergestellt. Für die Implementierung der Schnittstelle waren mehrere Stunden notwendig, da sie für JIRA noch nicht vorlag.

Der Usability Workshop fand am 19. und 20.09.2007 jeweils von 9:00 Uhr bis 18:00 Uhr im Schulungszentrum von TRUMPF statt. Die Organisation und Durchführung des Workshops wurde von der Moderatorin übernommen, während die Forscher die Erlaubnis hatten an beiden Tagen beobachtend teilzunehmen. Zu Beginn des Workshops demonstrierten die Forscher vor allen Teilnehmern kurz die Funktionalität von OpenProposal und standen während des Workshops für Fragen und Problemen zu OpenProposal zur Verfügung, ohne hierbei Einfluss auf den Testbetrieb zu nehmen.

Zur Datenerhebung wurden die Beobachtungen protokolliert, Fragebögen an die Anwender ausgeteilt, die erstellten Anwenderbeiträge ausgewertet und Befragungen mit der Moderatorin, dem Entscheider und einem Entwickler durchgeführt. Abschließend wurde ein Fallstudienbericht erstellt, der mit der Moderatorin abgestimmt und an alle Teilnehmer zur Durchsicht versendet wurde. Darauf aufbauend wurde bei TRUMPF die Entscheidung getroffen, dass OpenProposal zukünftig weiter für die Anwenderbeteiligung im Rahmen von Usability Workshops verwendet werden soll.

9.4.3 Ergebnisse der Fallstudie „TRUMPF“

Der Verlauf der Studie erfolgte gemäß dem zeitlichen Ablaufplan. Dabei konnte eine Vielzahl von Daten erhoben werden, anhand derer die Hypothesen zu OpenProposal überprüft wurden. Im Folgenden werden die Ergebnisse der Datenerhebung in Form von Beobachtungsprotokollen sowie anhand der erstellten Verbesserungsvorschläge, der ausgefüllten Fragebögen und der Experteninterviews zusammengefasst aufgeführt.

9.4.3.1 Auswertung der Beobachtungsprotokolle

Die Beobachtungen vor, während und nach dem Workshop wurden von den zwei Forschern handschriftlich festgehalten. Der erste Forscher war zuständig für die Beobachtung des gesamten Szenarios, während der zweite den Anwendern für Fragen oder Anmerkungen zu OpenProposal zur Verfügung stand. Letzterer betrachtete bei diesen Gelegenheiten zudem die Bildschirme und die vom jeweiligen Anwender genutzte Arbeitsweise. Hauptaugenmerk der Beobachtungen lag auf der Datenerhebung der Variablen AV5 „Komplexität der Erstellung von Anwenderbeiträgen“, AV6 „Zufriedenheit der Anwender mit der Methode zur Anwenderbeteiligung“, AV7 „Aufwand des Moderators für die Vorbereitung zur Erfassung von Anwenderbeiträgen“, AV8 „Aufwand des Moderators für die Nachbereitung der erstellen Anwenderbeiträge“ und AV9 „Aufwand des Moderators für die Interaktion mit den Anwendern“. Somit wurde der Aufwand zur Vor- und Nachbereitung sowie der Aufwand zur Erstellung und Erfassung der Anwenderbeiträge und die Zufriedenheit der Anwender untersucht.

Aus den erhobenen Beobachtungen zeichnete sich ein überwiegend positives Bild zu OpenProposal ab. Obwohl die Funktionalität von OpenProposal zu Beginn lediglich kurz demonstriert worden war, waren die Anwender mit dem Bedienkonzept und der Funktionsweise von OpenProposal innerhalb kurzer Zeit vertraut. Dabei fiel auf, dass sich die Art und Weise der Bedienung von OpenProposal der einzelnen Anwender stark voneinander unterschied. Während das Fenster von OpenProposal bei drei Personen ständig im Vordergrund geöffnet blieb, minimierten vier andere das Fenster und riefen es erst bei Bedarf auf. Zwei weitere Testpersonen beendeten OpenProposal nach jedem Absenden eines Beitrags. Der Internetbrowser mit JIRA wurde von allen Anwendern in den Hintergrund minimiert und erst beim Erstellen eines Beitrags in den Vordergrund geöffnet, wodurch die Testsoftware vollständig verdeckt wurde.

Während des Workshops suchten die Anwender mehrmals unaufgefordert das Gespräch mit den Forschern, wobei sie das Konzept der Annotation von Bildschirmfotos positiv hervorhoben. Als positiv wurden dabei die angenehme grafische Oberfläche, die Performanz und Bedienbarkeit der Annotationswerkzeuge und die Einfachheit der Erstellung von Anwenderbeiträgen hervorgehoben. Neben den positiven Äußerungen wurden zudem drei Verbesserungsvorschläge zu OpenProposal abgegeben, die teilweise in die späteren Weiterentwicklungen aufgenommen werden konnten.

So wurde erstens vorgeschlagen, eine Möglichkeit anzubieten das Bildschirmfoto in die Zwischenablage zu kopieren, um die Beiträge von OpenProposal auch direkt in Textverarbeitungsdokumente über die Zwischenablage einfügen zu können. Somit sollte OpenProposal beispielsweise auch für die technische Dokumentation eingesetzt werden können. Zweitens wurde gewünscht, dass Dateien an einen Vorschlag angefügt werden könnten. Dieser Wunsch entstand aus der Motivation heraus, dass einer der Anwender das Planungsdokument der Testsoftware einsenden wollte, um es dem Entwickler zu ermöglichen, die Situation des Anwenders in diesem Augenblick nachzuvollziehen. Drittens

wurde kritisch angemerkt, dass OpenProposal keine Unterstützung bei der Formulierung von Anwenderbeiträgen bietet, die einen Ablauf über mehrere Bereiche der Software bzw. mehrere Anwendungsfenster und Interaktionsdialoge betreffen. Hierzu reicht die Annotation eines einzelnen Bildschirmfotos nach Aussagen zweier Anwender nicht aus:

> *„Ich hätte gerne gezeigt, wie ich zu diesem Menü gekommen bin. Dafür müsste ich jetzt ja mit mehreren Vorschlägen arbeiten. Kann man da auch Screenshots zusammenstecken?“*

> *„Abläufe sind mit OpenProposal schlecht abzubilden.“*

Es konnte zudem beobachtet werden, dass einer der Teilnehmer Schwierigkeiten bei der Erstellung seiner Anwenderbeiträge hatte. Im Gespräch machte er folgende Aussage:

> *„Ich habe diesen Button angeklickt, keine Ahnung wie der heißt, der ist so klein, dass ich nicht erkenne was es bedeutet, da fällt mir nicht ein wie ich diesen textlich beschreiben soll.“*

Dies lässt darauf schließen, dass die Funktionsweise von OpenProposal nicht vollständig erfasst wurde: Der Anwender musste darauf hingewiesen werden, dass dieses Problem mit OpenProposal gelöst werden kann, indem der Vorschlag auf diesen „Button“ annotiert wird, wodurch eine weitere Bezeichnung überflüssig wird.

Zur Erstellung von Anwenderbeiträgen wurden neben OpenProposal und JIRA auch das Whiteboard und die Diskussion am Ende jedes Workshoptages genutzt. An das Whiteboard wurden im Laufe der beiden Tage mehrere Hinweise zu Problemen der Software geheftet. Jeder Hinweis bestand aus einem oder zwei Stichworten. Im Rahmen der Gruppendiskussion wurden die Meinungen der Teilnehmer ebenfalls mit Kärtchen abgefragt und die Bewertung der Software im Allgemeinen eingeholt. Zudem wurden die bereits notierten Hinweise am Whiteboard aufgegriffen, von ihren Verfassern kurz erläutert und in der Runde diskutiert. An der Diskussion nahmen die Anwender, die Moderatorin, der Entscheider und der Entwickler teil. Die Moderatorin notierte sich das Ergebnis der Diskussionen und gab die sich daraus ergebenden Verbesserungsvorschläge über JIRA in einen PC ein. Insgesamt war für die Durchführung der Diskussion und die Nachbereitung der Vorschläge über das Whiteboard ein Aufwand von ca. 120 Minuten notwendig.

In der Diskussion wurde - von den Forschern nicht initiiert - auch die Nützlichkeit von OpenProposal diskutiert. Alle Anwender zogen eine durchgängig positive Bilanz bezüglich OpenProposal. Ein Anwender machte hier folgende Aussage:

> *„OpenProposal erzeugt durch seine Einfachheit eine geringe Hemmschwelle welche dazu animiert, auch wirklich Rückmeldungen zu geben.“*

Die Moderatorin und der Entscheider äußerten sich ebenfalls sehr positiv über OpenProposal und über die Kombination von OpenProposal mit dem Issue-Tracker JIRA. Sie merkten an, dass OpenProposal im Rückblick auf frühere Workshops zu einer spürbaren Verbesserung geführt hatte und in Zukunft weiter eingesetzt werden soll.

Bezüglich des Aufwandes der Vor- und Nachbereitung konnten zudem weitere positive Effekte festgestellt werden. Bis auf die einmalige Integration von OpenProposal in JIRA war kein zusätzlicher Aufwand für die Vorbereitung notwendig. Auch die Nachbereitung der Anwenderbeiträge mithilfe von JIRA erforderte keinen zusätzlichen Aufwand. Sie erfolgte bereits während des Workshops. Die Moderatorin sichtete hierfür die in JIRA eingehenden Anwenderbeiträge, überarbeitete diese und ordnete sie den zuständigen Entwicklern zu. Der einzige Aufwand, der bei der Nachbereitung entstand, war die Anpassung der Titel, da diese häufig zu allgemein formuliert waren und keine konkrete Aussage enthielten. Für die Nachbereitung der mit OpenProposal und JIRA abgegebenen Beiträge waren etwa fünf Minuten erforderlich.

9.4.3.2 Auswertung der erstellten Anwenderbeiträge

Alle Anwenderbeiträge lagen nach Abschluss des Workshops in JIRA vor, so dass diese vor Ort bei TRUMPF gesichtet und analysiert werden konnten. Anhand der eingangs definierten Messkriterien bzw. der abhängigen Variablen AV1 „Anzahl der erstellen Anwenderbeiträge“, AV2 „Umfang der erstellten Anwenderbeiträge“, AV3 „Konstruktivität der erstellten Anwenderbeiträge“ und AV10 „Verständlichkeit der erstellten Anwenderbeiträge“ wurden bei der Auswertung die Anzahl, der Umfang, die Konstruktivität und die Verständlichkeit der erstellten Verbesserungsvorschläge ermittelt und darüber hinaus auf Auffälligkeiten geachtet.

Während des Workshops konnten insgesamt 71 Anwenderbeiträge erfasst werden: 30 mit OpenProposal, 14 mit JIRA und 27 auf den Kärtchen des Whiteboards. Eine Übersicht über die mit OpenProposal und JIRA abgegebenen Anwenderbeiträge und die Verteilung der abgegebenen Vorschläge auf die Anwender stellt Abbildung 9.4 dar.

Auf den Kärtchen des Whiteboards wurden vor allem allgemeine Kommentare zur Software abgegeben. Beispielsweise war zu lesen: „erster Eindruck positiv“, „insgesamt intuitiv“, „Formvorlagen?“, „Terminologie verbessern“, „Wie oft geht Redo-Undo?“, „Knopf Messer fehlt oder nicht zu finden“, „Gibt es ein Zwischenspeichern?“. Für die weitere Auswertung wurden die auf dem Whiteboard abgegebenen Vorschläge nicht betrachtet, da diese Vorschläge erst nachträglich von der Moderatorin ausführlich formuliert und nicht elektronisch vom Anwender erfasst wurden. Zudem wurden nur sechs der 27 Kärtchen während des Tests, alle anderen Kärtchen jedoch erst im Rahmen der Diskussionsrunde erstellt.

Die mit OpenProposal erstellten Anwenderbeiträge erhielten durchschnittlich eine Annotation mit einer aus 12,86 Wörter (MW: 12,86; Median: 10,50; MIN: 0,00; MAX: 34,00; STABW: 9,32) bestehenden erläuternden Beschreibung sowie einen Titel und eine allgemeine Kurzbeschreibung mit insgesamt 10,90 Wörtern (MW: 10,90; Median: 10,50; MIN: 0,00; MAX: 29,00; STABW: 6,37). Insgesamt wurden bei jedem OpenProposal Beitrag durchschnittlich 23,76 Wörter (MW: 23,76; Median: 23,50; MIN: 4,00; MAX: 58,00; STABW: 8,97) eingegeben. Die rein textbasierten JIRA Anwenderbeiträge besaßen einen aus durchschnittlich 18,64 Wörtern (MW: 18,64; Median: 16,00; MIN: 7,00; MAX: 49,00; STABW: 7,82) bestehenden Titel und Textkörper. Bei drei JIRA Beiträgen wurden ergänzend zum Text Bildschirmfotos angehängt.

Als Annotationswerkzeuge von OpenProposal wurde in 22 Fällen das „Kommentar“-Werkzeug, in sechs Fällen das „Verschieben“-Werkzeug und in einem Fall das „Hinzufügen“-Werkzeug verwendet. Bei einem Fall wurde der Vorschlag als reines Bildschirmfoto ohne eine Annotation bzw. Kategorisie-

rung versendet. Die anderen Annotationswerkzeuge wurden nicht verwendet. Eine Betrachtung der Inhalte der Annotationen ergab, dass von den 23 Annotationen, die als „Kommentar“ kategorisiert wurden, in 16 Fällen eine andere Kategorisierung besser geeignet gewesen wäre. Bei dem Anwenderbeitrag, der keine Annotation enthielt, handelte es sich um eine Fehlermeldung. Insgesamt waren 25 der 30 OpenProposal Anwenderbeiträge konstruktive Verbesserungsideen und die restlichen fünf Problemmeldungen. Die JIRA Anwenderbeiträge wurden von den Anwendern nicht kategorisiert. 10 der 14 JIRA Beiträge waren Problemberichte.

Bei der Sichtung der Anwenderbeiträge führten die Moderatorin und der Entwickler eine Bewertung der eingegangen Anwenderbeiträge durch. Als Bewertungsskala diente die Schulnotenskale von 1,00 für sehr gut bis 6,00 für ungenügend. Die mit OpenProposal erstellten Anwenderbeiträge wurden durchschnittlich mit der Note 1,67 (MW: 1,67; Median: 1,00; MIN: 1,00; MAX: 4,00; STABW: 0,71) bewertet. Die JIRA Anwenderbeiträge erzielten eine durchschnittliche Bewertung von 1,86 (MW: 1,86; Median: 2,00; MIN: 1,00; MAX: 4,00; STABW: 0,73). Sowohl bei den OpenProposal als auch bei den JIRA Beiträgen musste der Titel in jeweils drei Fällen nachträglich angepasst werden, da er keine konkrete Aussage enthielt. Anteilsmäßig mussten somit 10,00 % der OpenProposal Anwenderbeiträge und 21,43 % der JIRA Anwenderbeiträge überarbeitet werden. Insgesamt wurde die Formulierung der Titel häufig als ungeeignet beurteilt, da diese oftmals sehr allgemein formuliert und kurz gehalten wurden. Aus ihrer Sicht vergrößert der Titel jedoch die Übersicht bei der Bearbeitung einer großen Liste von Anwenderbeiträgen und spielt daher für die Moderatorin und den Entwickler eine wichtige Rolle. Die Moderatorin formulierte dabei folgenden Vorschlag für OpenProposal:

> *„Ein weiterer Umstand, der die Einordnung der Ergebnisse erschwert ist, dass viele Nutzer keinen Titel und/oder keine Beschreibung vergeben. Vielleicht wäre es möglich, OpenProposal so zu gestalten, dass der Vorschlag nur abgeschickt werden kann, wenn Titelzeile und Beschreibung ausgefüllt wurden?“*

Alle Anwender erstellten mit OpenProposal einen oder mehrere Anwenderbeiträge. Mit JIRA gaben lediglich drei Personen Anwenderbeiträge ab. Dies bedeutet, dass acht Anwender ausschließlich OpenProposal und nicht JIRA verwendet hatten.

Auch wenn aus diesen Daten keine statistisch haltbare Aussage abgeleitet werden kann und eine quantitative Analyse nicht angestrebt wurde, konnten bei den erstellten Anwenderbeiträge dennoch zahlreiche Auffälligkeiten identifiziert werden:

- Mit OpenProposal wurden mehr Anwenderbeiträge als mit JIRA abgegeben. Die Möglichkeit, Anwenderbeiträge auch mündlich abzugeben, wurde genauso häufig wie OpenProposal genutzt.
- Die Anzahl der Wörter war bei den mit OpenProposal eingereichten Anwenderbeiträgen höher als bei mit JIRA abgegebenen Anwenderbeiträgen.
- Die Kategorisierung der Anwenderbeiträge wurde nur bei OpenProposal angewendet. Allerdings wurden nicht das gesamte Spektrum und häufig auch nicht die richtige Kategorie ausgewählt. In JIRA wurden von den Anwendern keine Kategorisierungen vorgenommen.
- Die Verständlichkeit der mit OpenProposal und mit JIRA abgegebenen Anwenderbeiträge wurde von der Moderatorin und dem Entwickler als gut bewertet. Bei OpenProposal traten

anteilmäßig weniger unverständliche Anwenderbeiträge als bei JIRA auf. Außerdem mussten weniger Anwenderbeiträge nachträglich angepasst werden. Die Formulierung der Titel wurde bei beiden als schlecht bewertet.

OpenProposal wurde von allen elf Anwendern benutzt, JIRA jedoch nur von dreien.

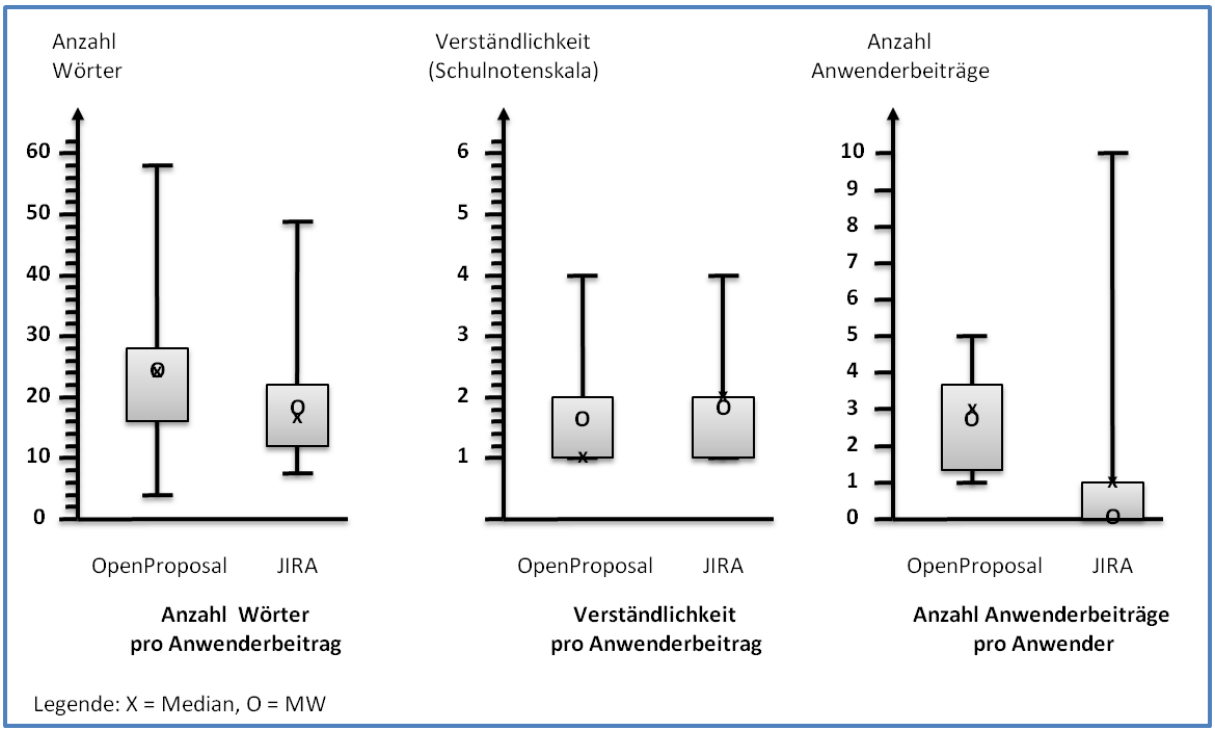

Abbildung 9.4: Übersicht über die abgegebenen Anwenderbeiträge in der Fallstudie „TRUMPF“

9.4.3.3 Auswertung der Fragebögen

Zur Erhebung der Akzeptanz durch den Anwender wurde ein Fragebogen gestaltet, der jedem Anwender am Ende des ersten sowie des zweiten Workshoptages ausgeteilt wurde. Die Forscher waren beim Ausfüllen der Fragebögen nicht im Raum anwesend und darüber hinaus um eine geringstmögliche Beeinflussung der Anwender bemüht Die Gestaltung des Fragebogens erfolgte nach den etablierten Richtlinien der sozialwissenschaftlichen Forschung (vgl. [Schnell et al., 2008]) und wurde vor Beginn des Workshops mehrfach an Testpersonen außerhalb von TRUMPF getestet. Mit dem Fragebogen wurden hauptsächlich Daten zu den Variablen AV5 „Komplexität der Erstellung von Anwenderbeiträgen“ und AV6 „Zufriedenheit der Anwender mit der Methode zur Anwenderbeteiligung“ erhoben. Es wurde auch die Möglichkeit zur Abgabe von allgemeinen Kommentaren geboten, so dass auch Hinweise zu anderen Variablen erfasst werden konnten.

Der Fragebogen gliederte sich in sechs Teile. Mit dem Deckblatt wird der Anwender über die Ziele und Aufbau des Fragebogens und über die Vorgehensweise zur Anonymisierung informiert. Außerdem werden wichtige Hinweise zum Ausfüllen des Fragebogens gegeben, u.a. dass in der Fußzeile jeder Seite des Fragebogens zusätzliche allgemeine Kommentare zum Fragebogen abgegeben werden können. Abschließend wird den Anwendern für ihre Bemühungen bedankt.

In zweiten Teil werden einleitend allgemeine Angaben zur Person erfragt. Hierzu gehören Fragen zur Berufserfahrung, zum Geschlecht, zur Tätigkeit bzw. Fachrichtung und zu testrelevanten Handicaps.

Ferner wird die Testperson nach ihren bisherigen Erfahrungen zur Formulierung von Anforderungen befragt und um eine Selbsteinschätzung ihrer Fähigkeit bei der Formulierung von Anforderungen gebeten. Außerdem werden Fragen zur Vertrautheit mit der getesteten Software gestellt.

In den vier weiteren Teilen des Fragebogens wird der Anwender zur Einschätzung der Gebrauchstauglichkeit und Nützlichkeit von OpenProposal und JIRA befragt. Dabei konnte der Anwender seine Einschätzung zu vorformulierten Thesen auf einer Likert-Skala mit sieben Merkmalsausprägungen abgeben. Als Antwortmöglichkeiten standen zu jeder These die Ausprägungen „Trifft überhaupt nicht zu“, „Trifft größtenteils nicht zu“, „Trifft eher nicht zu“, „Weder noch“, „Trifft eher zu“, „ Trifft größtenteils zu“ und „Trifft voll und ganz zu“ zur Auswahl. Außerdem konnte der Anwender sich einer Antwort enthalten bzw. mit „Ist mir nicht aufgefallen“ antworten.

Um der Akquieszenz vorzubeugen, also der Tendenz, den vorgegebenen Thesen unabhängig von ihrem Inhalt zuzustimmen, wurden die Thesen abwechselnd positiv und negativ für das jeweilige Werkzeug formuliert. Die Anwender wurden gebeten, den dritten und vierten Teil nur auszufüllen falls sie an diesem Tag OpenProposal verwendet, und den fünften Teil nur auszufüllen, falls sie JIRA verwendet hatten. Der sechste Teil sollte von allen erst zum Schluss des Workshops beantwortet werden, falls sie im Workshop beide Werkzeuge über beide Tage verwendet hatten.

Die Thesen im dritten Teil beziehen sich auf die Darstellung der grafischen Benutzeroberfläche, auf die Hilfestellungen im Programm, auf die Bedienbarkeit und auf den Reifegrad von OpenProposal. Im vierten bzw. fünften Teil werden Thesen zur Nützlichkeit von OpenProposal bzw. JIRA aufgelistet. Die Thesen wurden zu jeder Methode identisch gehalten, so dass sie sich einzig in der Bezeichnung des eingesetzten Methode OpenProposal bzw. JIRA unterschieden. Im sechsten Teil wurden vergleichende Thesen zu OpenProposal und JIRA aufgestellt, um den Anwender beide Methoden direkt miteinander vergleichen zu lassen.

Von den elf ausgeteilten Fragebögen wurden acht Fragebögen vollständig und auswertbar ausgefüllt. Tabelle 9.4, Tabelle 9.5 und Tabelle 9.6 listen die wichtigsten Thesen zu OpenProposal, JIRA und zu beiden im Vergleich auf und fasst die Antworten mit dem ermittelten Mittelwert (MW), der Varianz (VAR) sowie Minimum (MIN) und Maximum (MAX) zusammen. Die Antworten sind folgendermaßen kodiert: „Enthaltungen = Ist mir nicht aufgefallen“, „1,00 = Trifft überhaupt nicht zu“, „2,00 = Trifft größtenteils nicht zu“, „3,00 = Trifft eher nicht zu“, „4,00 = Weder noch“, „5,00 = Trifft eher zu“, „6,00 = Trifft größtenteils zu“ und „7,00 = Trifft voll und ganz zu“.

Die Bewertungen der Anwender zu OpenProposal (vgl. Tabelle 9.4) fielen im Durchschnitt sehr positiv aus. Die Antworten weisen darauf hin, dass die Anwender mit der Bedienung von OpenProposal gut zurechtkamen und Vorschläge schnell und unkompliziert abgegeben werden konnten. Die Symbolik und Namensgebung und die Einfachheit der Erstellung von Anwenderbeiträgen ist im Vergleich zu den anderen Aspekten verbesserungswürdig. Die Varianz zwischen den Antworten aller Anwender war bis auf wenige Ausnahmen sehr niedrig.

Die Antworten in Tabelle 9.5 bezeugten auch eine positive Wahrnehmung von JIRA, wobei zu einigen Thesen im Vergleich zu OpenProposal deutlich schlechtere Bewertungen abgegeben wurden. Bis auf das schnelle Zurechtfinden wurde JIRA in allen Aspekten weniger gut aber dennoch positiv bzw. neutral bewertet. Die größten Abweichungen finden sich in der Informationsdarstellung und der

Einfachheit der Erstellung von Anwenderbeiträgen, bei der OpenProposal eindeutig besser eingeschätzt wurde.

Thesen	MW	VAR	MIN	MAX
Ich habe mich schnell in OpenProposal zurecht gefunden	6,25	0,21	6,00	7,00
OpenProposal zeigt zu viel Information auf einmal; das hat mich verwirrt.	1,25	0,21	1,00	2,00
Symbole und Namensgebung in OpenProposal sind eingängig.	5,29	0,24	5,00	6,00
Das Erstellen von Vorschlägen war unnötig kompliziert und langwierig.	1,13	0,13	1,00	2,00
Vorschläge sind einfach und ohne viel Nachdenken zu müssen zu erstellen.	5,75	0,21	5,00	6,00
Ich bin bei OpenProposal oft stecken geblieben und musste einen anderen Weg suchen.	1,38	1,13	1,00	4,00

Tabelle 9.4: Antworten zu den Thesen zu OpenProposal im Fragebogen der Fallstudie „TRUMPF“

Thesen	MW	VAR	MIN	MAX
Ich habe mich schnell in JIRA zurecht gefunden	6,20	0,70	5,00	7,00
JIRA zeigt zu viel Information auf einmal; das hat mich verwirrt.	2,60	1,80	2,00	5,00
Symbole und Namensgebung in JIRA sind eingängig.	4,00	1,20	3,00	6,00
Das Erstellen von Vorschlägen war unnötig kompliziert und langwierig.	2,67	2,67	1,00	5,00
Vorschläge sind einfach und ohne viel Nachdenken zu müssen zu erstellen.	5,00	1,60	3,00	6,00
Ich bin bei JIRA oft stecken geblieben und musste einen anderen Weg suchen.	1,86	0,14	1,00	2,00

Tabelle 9.5: Antworten zu den Thesen zu JIRA im Fragebogen der Fallstudie „TRUMPF“

Im direkten Vergleich in Tabelle 9.6 zwischen OpenProposal und JIRA wurde eindeutig zugestimmt, dass die Erstellung eines Anwenderbeitrags mit OpenProposal einfacher ist als mit JIRA und die Anwender die Nutzung von OpenProposal bevorzugen würden. Die Anwender würden eine Kombination von OpenProposal mit JIRA favorisieren. JIRA als alleinige Lösung wird nicht gewünscht.

Neben den Verbesserungsvorschlägen, die schon in den Gesprächen während den Tests geäußert wurden, gaben die Anwender in den Kommentarfeldern im Fragebogen weitere Anmerkungen ab. Als Vorteile von OpenProposal wurde hervorgehoben, dass die Einfachheit zur Erstellung von An-

wenderbeiträgen und die nahtlose Integration in JIRA eine Motivation zur Erstellung von Anwenderbeiträgen darstellt:

> *„OpenProposal ist enger [als JIRA] gefasst aber dadurch einfacher einsetzbar. OpenProposal [dient somit] als Eingangstor zu JIRA.“*

Ein anderer Anwender resümiert über die Vor- und Nachteile textlicher Formulierungen und kommt zu dem Schluss, dass mit einer textlichen Beschreibung jede Thematik erfasst werden kann, die Formulierungen für manche Fälle allerdings auch kompliziert ausfallen können:

> *„Textuelle Vorschläge können alle Themen abdecken, aber vor allem im graphischen Bereich ist es manchmal sehr aufwendig zu beschreiben was man meint. Sie sind also breiter aufgestellt, aber komplizierter in manchen Fällen.“*

Thesen	**MW**	**VAR**	**MIN**	**MAX**
Ich würde den grafischen Ansatz von OpenProposal dem textbasierten Ansatz von JIRA vorziehen.	5,62	0,55	4,00	6,00
Ich würde JIRA eher einsetzen als OpenProposal.	4,20	0,20	4,00	5,00
Ich finde die Vorschlagserstellung bei OpenProposal einfacher als bei JIRA.	6,29	0,57	5,00	7,00
Ich finde die Vorschlagserstellung bei OpenProposal mit weniger Aufwand verbunden als bei JIRA.	5,50	1,67	4,00	7,00
Für mich käme nur eine Kombination von JIRA und OpenProposal in Frage, wenn es um die Vorschlagserstellung geht.	7,00	0,00	7,00	7,00

Tabelle 9.6: Antworten zu den vergleichenden Thesen im Fragebogen der Fallstudie „TRUMPF“

Die Gesamtbewertung zu OpenProposal und JIRA fiel schließlich eindeutig aus. Während JIRA im durchschnittlich mit der Note 2,43 (MW: 2,43; Median: 3,00; MIN: 1,00; MAX: 3,00; VAR: 0,62) bewertet wurde, erzielte OpenProposal die Note 1,88 (MW: 1,88; Median: 2,00; MIN: 1,00; MAX: 2,00; VAR: 0,13). Somit wurde OpenProposal auf der Schulnotenskala deutlich besser als JIRA bewertet.

9.4.3.4 Auswertung der Experteninterviews

Nach Abschluss des Entwicklungsprojektes der TRUMPF Software, die in dem Usability Workshop getestet wurde, wurden separate Befragungen mit der Moderatorin, dem Entscheider und einer Entwicklerin des Entwicklerteams durchgeführt. In diesem Zeitraum wurde OpenProposal in fünf weiteren Usability Workshops bei TRUMPF eingesetzt, die allerdings aus Gründen der Geheimhaltung den Forschern nicht für weitere Untersuchungen zugänglich gemacht wurden.

Für die Befragungen wurde ein Leitfaden erstellt, um ihnen eine grobe Struktur vorzugeben und dabei die Gespräche nicht zu stark einzuschränken bzw. einzugrenzen. Der Leitfaden wurde aus den Variablen AV2, AV3, AV4, AV5, AV6, AV7, AV8, AV9 und AV10 abgeleitet und enthielt Fragen zum

Aufwand der Vor- und Nachbereitung sowie Durchführung der Workshops, der Verständlichkeit und Konstruktivität der Vorschläge und zur Zufriedenheit mit den Ergebnissen des Usability Workshops. Jedes Gespräch in diesen Befragungen wurde aufgezeichnet und ein Mitschrieb angefertigt, auf dessen Basis die Auswertung angefertigt wurde.

Die Moderatorin sah in OpenProposal den klaren Vorteil, dass die Vor- und Nachbereitung der Workshops entfällt. Die Verständlichkeit und Aussagekraft der Anwenderbeiträge bewertete sie als sehr gut. Mit dem Ablauf des Workshops war sie sehr zufrieden und befand OpenProposal für hilfreich. Die Kombination aus JIRA, OpenProposal und den Kärtchen hielt sie bestens geeignet für die Anwenderbeteiligung bei TRUMPF. Diese Erfahrungen ließen sich in den nachfolgenden Workshops bestätigen, so dass der Wunsch ausgesprochen wurde OpenProposal weiter zu verwenden. Es war auch die Idee entstanden, OpenProposal im Sinne einer neuen Dienstleistung für die Kommunikation mit den Kunden einzusetzen, um auch von deren Anwendern Verbesserungsvorschläge zu erhalten. Den einzigen Schwachpunkt stellte die Kategorisierung der Anwenderbeiträge dar, da die Anwender nach Einschätzung der Moderatorin mit der Auswahl des geeigneten Annotationswerkzeuges Schwierigkeiten hatten:

> *„Was mir beim letzten Workshop im Oktober noch aufgefallen ist: Die Tester machen sich nicht immer die Mühe zu schauen, welche der bei OpenProposal zur Auswahl stehenden Kommentarkategorien die jeweils angemessenste ist."*

Andere Methoden wie Videoanalysen und Papierprototypen hielt sie für ihre Arbeit nicht geeignet. Allerdings unterschied sie zwischen Funktionalitäts- und Anwendertests:

> *„Bei reinen Funktionalitätsworkshops, in denen getestet wird, ob alle neu eingebauten Funktionen korrekt funktionieren und ob die neue Version insgesamt stabil läuft, sind die Tester überwiegend Entwickler und es werden überwiegend sogenannte Testfälle abgearbeitet, die als JIRA-Punkte vorliegen. JIRA ist bei uns in der Software-Entwicklung das System zur Verwaltung aller Arbeitspunkte und wird von uns Entwicklern täglich eingesetzt. Das heißt, es sind Personen, die tagtäglich JIRA benutzen und damit sehr vertraut sind. Und JIRA ist sowieso schon an der richtigen Stelle, dem jeweiligen Testfall, geöffnet, so dass gleich die Bewertung, i.d.R. ‚ok', eingetragen werden kann."*

Somit eignete sich aus ihrer Sicht JIRA besser für Funktionalitätstests als OpenProposal, da die Entwickler JIRA als Methode für das Aufgaben- und Testmanagement verwenden und die Tests somit in JIRA dokumentieren können. Dies gilt jedoch nur für die abteilungsinternen Entwickler, die mit JIRA vertraut sind. Für reale Anwender, die JIRA sehr selten verwenden, ist mit OpenProposal besser geeignet:

> *„JIRA hat allerdings hinsichtlich von Workshops zwei Schwachstellen: 1. Es ist komplex und daher für Personen, die eher selten damit umgehen, nicht so einfach zugänglich. Und 2. ist das Erstellen von Screenshots etwas umständlich. Bei Usability Workshops dagegen sind mehr Personen beteiligt, die nur wenig mit JIRA zu tun haben. Außerdem sind viele Kommentare zu Oberflächenelementen erforderlich, so dass Screenshots gemacht werden müssen. Beides spricht für OpenProposal als einfaches Screenshot-Eingangsportal zu JIRA."*

Der Entscheider nannte als Vorteil von OpenProposal, dass die Anwender ihre Beiträge im Vergleich zu JIRA unkompliziert und schnell abgeben konnten. Die Anwenderbeiträge, die mit OpenProposal erstellt wurden, waren aus seiner Sicht verständlich und nachvollziehbar. Er fand es besonders förderlich, dass die meisten Anwenderbeiträge konkret beschrieben wurden. Für ihn war es zudem hilfreich, dass er sich die Anwenderbeiträge bereits zum Ende des Workshops anschauen konnte und Entscheidungen zeitnah treffen konnte. Insgesamt war er sehr zufrieden mit dem Ablauf des Workshops und bekräftigte einen zukünftigen Einsatz von OpenProposal.

Die Entwicklerin hielt den Einsatz von OpenProposal für sinnvoll und war insbesondere mit der Konstruktivität der Anwenderbeiträge zufrieden. Den Aufwand der Vorbereitung der Workshops konnte sie nicht bewerten, da sie vorher auch keinen bemerkenswerten Aufwand hatte. Die Nachbereitung war dank OpenProposal und JIRA spürbar geringer, da das Abtippen der Papierbögen entfiel. Einziges Problem war, dass manche Anwenderbeiträge fehlende bzw. nicht aussagekräftige Titel besaßen. Dies betraf OpenProposal wie JIRA gleichermaßen. Bei JIRA mussten die Entwickler bei drei der 15 Anwenderbeiträge zusätzliche Informationen oder Änderungen nachtragen. Bei OpenProposal waren es drei Änderungen bei insgesamt 30 Vorschlägen. Daher äußerte die Entwicklerin den Wunsch, dass OpenProposal bei leerem Titel eine Bestätigungsabfrage ähnlich wie beim Versand einer E-Mail ohne Betreffzeile tätigen soll. Diese Option sollte per Konfiguration deaktiviert werden können und bei Bedarf auch auf ein leeres Beschreibungsfeld ausgedehnt werden können.

9.4.4 Diskussion der Ergebnisse der Fallstudie „TRUMPF“

Im Gesamten wurde OpenProposal von allen Beteiligten positiv bewertet. Von den Anwendern wurde OpenProposal als unkomplizierte und einfache Methode wahrgenommen. Die Moderatorin und der Entscheider sahen den großen Vorteil, dass die Nachbereitung der Workshops stark reduziert und die Entwicklerin befand die mit OpenProposal erstellten Anwenderbeiträge für verständlich.

Die Anwender verinnerlichten das Bedienkonzept von OpenProposal in kurzer Zeit. Die grafische Benutzeroberfläche und das Konzept der Annotation von Bildschirmfotos wurden als sehr gut bewertet. Es wurde mit einer großen Mehrheit einer zukünftig kombinierten Verwendung von JIRA und OpenProposal zugestimmt. Bei einem Teilnehmer wurde zu Beginn festgestellt, dass das Bedienkonzept nicht verstanden wurde. Dies konnte durch eine kurze Erklärung behoben werden. Als Verbesserungsideen zu OpenProposal wurde vorgeschlagen, das OpenProposal Annotationssystem um weitere Funktionen zu erweitern, um das annotierte Bildschirmfoto in die Zwischenablage kopieren, Dateien zum Vorschlag anhängen und Abläufe über mehrere Software-Bereiche in einem Vorschlag annotieren zu können.

Die Analyse der abgegebenen Anwenderbeiträge ergab, dass die OpenProposal Anwenderbeiträge zwar durchgängig kategorisiert wurden, aber die gewählten Kategorien häufig nicht korrekt waren, da die Anwender damit überfordert waren. Die Analyse ergab weiter, dass die mit OpenProposal erstellten Anwenderbeiträge mehr Wörter enthielten als diejenigen, die mit JIRA erstellt wurden. Somit ist insgesamt ein höherer Umfang der OpenProposal Anwenderbeiträge festzustellen. Dies kann damit begründet werden, dass OpenProposal den Anwender zu ausführlicheren und konkreteren Beschreibungen motiviert. Dies würde sich auch damit decken, dass den OpenProposal Anwenderbeiträge eine hohe Konstruktivität zugeschrieben wurde. Bei den OpenProposal Anwenderbeiträgen lag der Anteil der Problemberichte bei 16,67% (5 von 30), bei JIRA bei 71,43% (10 von 14). Somit

wurden mit OpenProposal deutlich mehr konstruktive Anwenderbeiträge abgegeben. Auffällig war auch, dass alle Anwender mit OpenProposal gearbeitet und nur drei der elf Anwender mit JIRA Anwenderbeiträge erstellt hatten. Die Verständlichkeit der Anwenderbeiträge - sowohl bei JIRA als auch bei OpenProposal - wurde als hoch bewertet. Für die Entwicklerin stellte sich dabei das Problem, dass einige Titel der Anwenderbeiträge nicht aussagekräftig waren und angepasst werden mussten.

AV	**UV1-1 OpenProposal**	**UV1-2 JIRA**
AV1: Anzahl der erstellten Anwenderbeiträge	30	14
AV2: Umfang der erstellten Anwenderbeiträge	Mittel 23,76 Wörter pro Vorschlag und 30 Bildschirmfotos	Mittel 18,64 Wörter pro Vorschlag und 3 Bildschirmfotos
AV3: Konstruktivität der erstellten Anwenderbeiträge	Hoch 25 konstruktiv 5 Fehlermeldungen	Niedrig 4 konstruktiv 10 Fehlermeldungen
AV4: Aufwand des Anwenders zur Erstellung von Anwenderbeiträgen	Niedrig	Mittel
AV5: Komplexität der Erstellung von Anwenderbeiträgen	Niedrig	Hoch
AV6: Zufriedenheit der Anwender mit der Methode zur Anwenderbeteiligung	Hoch 1,88	Mittel 2,43
AV7: Aufwand des Moderators für die Vorbereitung zur Erfassung von Anwenderbeiträgen	Niedrig	Niedrig
AV8: Aufwand des Moderators für die Nachbereitung der erstellten Anwenderbeiträgen	Niedrig 10,00% Änderungen	Niedrig 21,43% Änderungen
AV9: Aufwand des Moderators für die Interaktion mit den Anwendern	Niedrig	Niedrig
AV10: Verständlichkeit der erstellten Anwenderbeiträge	Hoch 1,67: Trifft größtenteils zu	Hoch 1,85: Trifft größtenteils zu

Tabelle 9.7: Zusammenfassung der Ergebnisse aus der Fallstudie „TRUMPF“

In Bezug auf die abhängigen Variablen AV1 - AV10 konnte in der Fallstudie eine aussagekräftige Datenbasis geschaffen werden, mit denen Antworten auf die Hypothesen H1 - H5 gegeben werden

können. Tabelle 9.7 setzt die Ergebnisse der Fallstudie in Bezug zu den abhängigen Variablen AV1 - AV10 und den unabhängigen Variablen UV1 mit OpenProposal und JIRA.

Im direkten Vergleich zu JIRA schneidet OpenProposal überwiegend besser ab. Für die Untersuchung der Hypothesen konnten dabei folgende Feststellungen gemacht werden:

1. Mit über doppelt so vielen Anwenderbeiträgen bei OpenProposal kann Hypothese H1 unterstützt werden.

2. Hypothese H2 kann unterstützt werden, da den mit OpenProposal erstellten Anwenderbeiträgen eine sehr hohe Konstruktivität zugesprochen wurden.

3. Hypothese H3 kann soweit zugestimmt werden, dass Anwender mit OpenProposal motiviert wurden, Anwenderbeiträge zu erstellen ohne dabei aktiv begleitet zu werden. Allerdings war kein realer Einsatz im Betriebsalltag möglich.

4. Im Vergleich zu den früheren Usability Workshops war im Workshop mit OpenProposal und JIRA nur ein geringer Aufwand zur Vor- und Nachbereitung notwendig. Dies unterstützt Hypothese H4.

5. Die Verständlichkeit der Anwenderbeiträge mit OpenProposal wurde als gut und geringfügig besser als die Verständlichkeit der Anwenderbeiträge mit JIRA bewertet. Dies unterstützt Hypothese H5.

In der ersten Fallstudie „TRUMPF“ konnte somit nachgewiesen werden, dass vier der fünf Hypothesen unterstützt werden können. Das Unternehmen TRUMPF setzte OpenProposal anschließend in weiteren Usability Workshops für den Test mit Anwendern weiter ein. Die erfassten Anwenderbeiträge zu OpenProposal wurden im Entwicklungsteam von OpenProposal diskutiert und verworfen, da diese die Komplexität von OpenProposal erhöht hätten und nur für spezielle Einzelfälle praktikabel gewesen wären. Einzig die Kategorien der Annotationswerkzeuge bedurften einer optischen Überarbeitung, da diese vom Anwender nicht korrekt angewendet wurden.

Die Fallstudie zeigte darüber hinaus auch Limitierungen von OpenProposal. Anwenderbeiträge, die sich über mehrere Anwendungsfenster erstrecken, sind mit OpenProposal nur sehr erschwert zu formulieren. Außerdem wird immer eine grafische Repräsentation in Form eines Prototypen oder einer funktionslosen grafischen Oberfläche der zu verbessernden Software benötigt. Insgesamt ist OpenProposal nur für die Einsatzszenarien sinnvoll, bei denen Anwender die grafische Bedienoberfläche einer Software an einem Computer testen können und bei denen in erster Linie konkrete und konstruktive Anwenderbeiträge gewünscht sind.

Im Anschluss an die erste Fallstudie wurde mit der Planung der Fallstudie „WAVES“ begonnen, in der die Ergebnisse aus der ersten Fallstudie repliziert und die Hypothese H3 unter realen Bedingungen im Betriebsalltag untersucht werden sollte.

9.5 Die Fallstudie „WAVES“[29]

Die Fallstudie „WAVES“ stellt die zweite Fallstudie zur Evaluation von OpenProposal dar. Im Folgenden werden zu Beginn der Projektkontext vorgestellt, dann das Design der Studie präsentiert und abschließend die Ergebnisse zusammengefasst und diskutiert.

9.5.1 Hintergrund zur Fallstudie „WAVES“

Das Verbundprojekt „WAVES – Wissensaustausch bei der verteilten Entwicklung von Software“ ist ein im Rahmen der Forschungsoffensive „Software Engineering 2006“ durch das BMBF gefördertes Forschungsprojekt, das im März 2006 startete und im Oktober 2008 endete. Das Projektkonsortium bestand aus den Technologiepartnern FZI Forschungszentrum Informatik[30], Empolis[31] und Polarion[32], den Anwendungspartnern CAS[33], PTV[34], Disy[35] und Object International[36] und dem Evaluationspartner Institut für Informatik der Freien Universität Berlin.

Ziel des Verbundprojekts war es, in verteilten Software-Projekten den Aufbau und den Austausch von informellem Wissen zu fördern und seine schrittweise Strukturierung und Vernetzung zu unterstützen. Hierbei sollten auch die Integration verschiedener überlappender Wissenssphären (persönliches Wissen, organisationsinternes bzw. team- oder projektbezogenes Wissen, öffentliches und allgemein zugängliches Wissen) unterstützt werden. Um das Ziel zu erreichen, verfolgte WAVES einen Lösungsansatz auf zwei Ebenen: der methodischen und der technischen Ebene.

---

[29] Herrn Herbert Schäfler ist an dieser Stelle zu danken, der als studentischer Mitarbeiter maßgeblich zur erfolgreichen Durchführung der Fallstudie beigetragen haben. Außerdem gebührt ein besonderer Dank der Moderatorin der FU Berlin und den Mitarbeitern der teilnehmenden Unternehmen.

[30] Das FZI Forschungszentrum Informatik ist eine Forschungseinrichtung des Landes Baden-Württemberg mit Geschäftssitz in Karlsruhe. Im Jahr 2008 beschäftigte das FZI 135 Mitarbeiter und hatte einen Umsatz von 11,3 Mio. € (Stand 01.03.2009).

[31] Empolis ist ein Software-Unternehmen mit Hauptsitz in Gütersloh und ein Tochterunternehmen der Avarto AG. Im Jahr 2008 hatte Empolis 230 Mitarbeiter und einem Umsatz von 30 Mio. € (Stand 31.03.2009)

[32] Polarion ist ein Software-Unternehmen mit Hauptsitz in Zürich. Im Jahr 2008 beschäftigte Polarion 20 Mitarbeiter (Stand 01.08.2008).

[33] CAS ist ein Software-Unternehmen mit Hauptsitz in Karlsruhe. Im Jahr 2007 beschäftigte CAS 178 Mitarbeiter mit ca. 33 Mio. € (Stand: 01. Oktober 2008).

[34] PTV ist ein Software-Unternehmen mit Hauptsitz in Karlsruhe mit 494 Mitarbeitern und 59,6 Mio. € Umsatz im Jahr 2007 (Stand: 01. Oktober 2008).

[35] Disy ist ein Software-Unternehmen mit Geschäftssitz in Karlsruhe und beschäftigte 2008 25 Mitarbeiter (Stand: 01. Oktober 2008).

[36] Object International ist ein Software-Unternehmen mit Geschäftssitz in Stuttgart mit ca. 10 Mitarbeitern im Jahr 2008 (Stand: 01. Oktober 2008).

Auf technischer Ebene wurden u.a. die Software-Werkzeuge WavesIS (WAVES Integrierte Suche), KISSy (Knowledge based Investigation of Software Systems) und Wiquila (Rich-Wiki-Client zur Unterstützung der schrittweisen Wissensartikulation) am FZI Forschungszentrum Informatik entwickelt, die zur Verbesserung des Wissensaustausches in Unternehmen beitragen sollen. WavesIS (vgl. Abbildung 9.5) ist eine Suchmaschine, mit der gleichzeitig in verschiedenen Informationsquellen wie z.B. Wikis, Versionsverwaltungssystemen, etc. gesucht werden kann. KISSy ist Teil der WavesIS und stellt eine Code-Suche bereit. Die von KISSy gefundenen Ergebnisse werden entgegen den in üblichen Code-Suchmaschinen gefundenen Treffer zusätzlich bewertet und dementsprechend sortiert. Wiquila ist ein Wiki-Editor, welcher u.a. an mehrere Wikis angebunden werden kann, über eine WYSISWG-Ansicht verfügt und eine Autovervollständigen-Funktion anbietet.

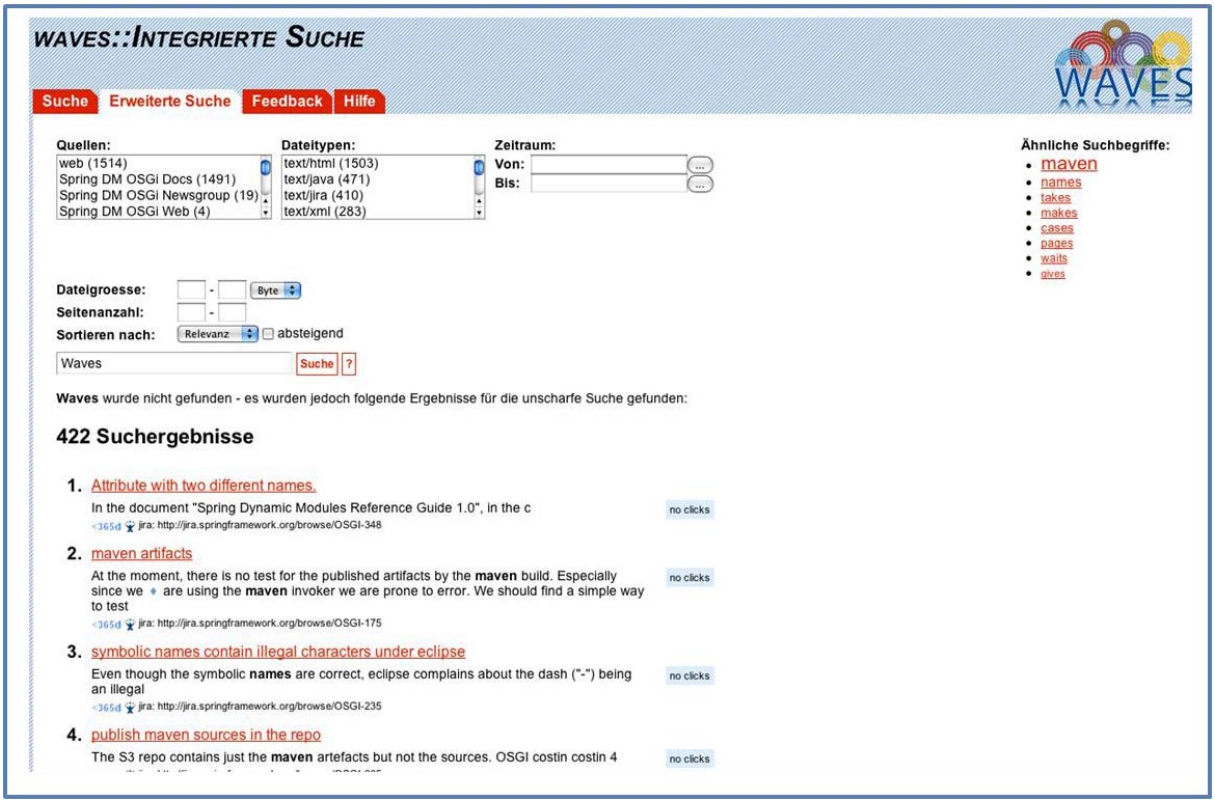

Abbildung 9.5: Bildschirmfoto von WavesIS

Auf methodischer Ebene wurde ein empirischer Ansatz gewählt, welcher die Ergebnisse der technischen Entwicklungsarbeit, der Anwendungsanalyse und der sozio-ökonomischen Untersuchungen verbindet. Die Arbeitsgruppe Software Engineering (AGSE) an der Freien Universität Berlin war als Evaluationspartner für die empirischen Analysen zuständig. Die Mitarbeiter der Anwendungspartner CAS, PTV, Disy und Object International waren hierfür als Anwender im Projekt beteiligt und erprobten die von den Technologiepartnern entwickelten Software-Werkzeuge.

Die Entwicklung und Evaluation in WAVES erfolgte nach einer iterativen Vorgehensweise (vgl. Abbildung 9.6) in mehreren Entwicklungs- und Evaluationsphasen. Bei Abschluss einer Entwicklungsphase wurde das Ergebnis der Phase als sogenannter Major Release in Form eines Prototypen zur Evaluation freigegeben. Bei den Anwendungspartnern wurde das Release installiert und erprobt. Der Evaluationspartner war anschließend für die Erfassung des Feedbacks der Anwendungspartner und der

Kommunikation an die Entwicklungspartner zuständig. Ziel der Evaluation war die kontinuierliche Verfeinerung der Anforderungen sowie die Bewertung des Nutzens des Systems. Dafür wurden für jedes Release in den Firmen Studien mit den Anwendern vor Ort durchgeführt. Die Fortschreibung der Anforderungen beinhaltete das Erfassen von Verbesserungsvorschlägen zur Benutzbarkeit und zur Funktionalität. Außerdem wurden Diskussionen und Beurteilungen zu den nächsten zu entwickelnden Features angeregt. Die Bewertung des Nutzens umfasste die Identifikation von Einsatzszenarien von WAVES in den Unternehmen und die Identifikation von ökonomischen, psychologischen und technischen Barrieren sowie deren Überwindung durch die WAVES Werkzeuge.

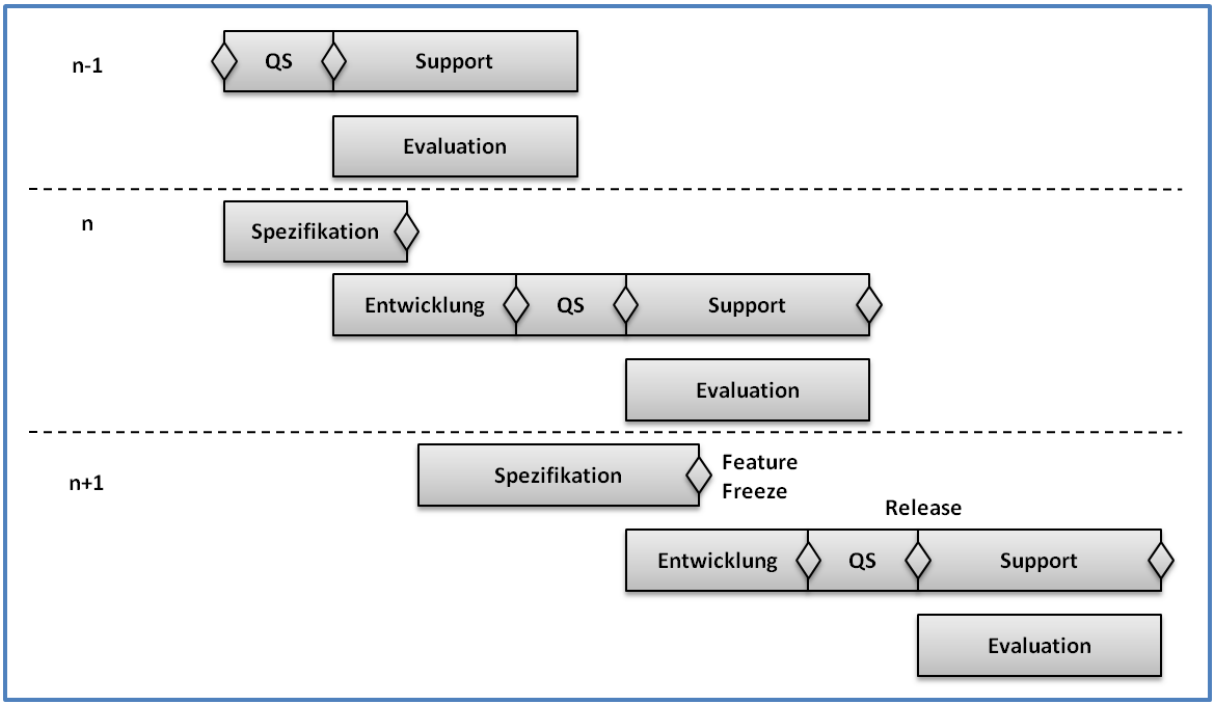

Abbildung 9.6: Iterative Vorgehensweise im Projekt WAVES

In den ersten zwei Evaluationsphasen wurden Einzelinterviews zur Erfassung von Verbesserungsvorschlägen durchgeführt, bei denen die Anwender vorgegebene Aufgaben mit den WAVES-Prototypen abarbeiten sollten. Die Anwender wendeten die „Thinking aloud"-Methode an und wurden dabei auf Video aufgenommen. Die Auswertung der Befragungen erfolgte mithilfe einer Videoanalyse. Die Moderatorin sah sich alle Videoaufnahmen aufmerksam durch und notierte die mündlich ausgesprochenen Verbesserungsvorschläge der Anwender in einem elektronischen Textdokument. Bei Vorschlägen, die bereits von anderen Anwendern genannt wurden, erhöhte sie den Zähler bei den entsprechenden Vorschlägen. Die Ergebnisse trug sie anschließend entweder im Projekt-Wiki oder in den Projekt-Issue-Tracker JIRA von Atlassian ein. Für die Kommunikation und das Aufgabenmanagement zwischen Moderatorin, Entscheidern und Entwickler wurden das Projekt-Wiki und JIRA verwendet. Zudem hatten auch alle Anwender der vier Anwenderunternehmen, die in WAVES bei den Usability Workshops beteiligt waren, darauf Zugriff. Sie konnten somit außerhalb der Evaluationsphase Verbesserungsvorschläge eingeben und den Verlauf der Bearbeitung ihrer Vorschläge verfolgen. In den regelmäßig stattfindenden Projekttreffen wurden zudem die Ergebnisse der Usability Workshops und der Stand der Überarbeitungen vorgestellt und darauf aufbauend die weiteren Schritte der Entwicklungen diskutiert.

Im Projekt waren zum Zeitpunkt des Beginns der Fallstudie im Juli 2008 bereits 28 von 36 Monaten vergangen. Die Prototypen befanden sich in der dritten Iterations-Phase „Major Release 2 (MR2)“. „Major Release 0 (MR0)” und „Major Release 1 (MR1)” waren bereits durchlaufen, wobei in den ersten beiden Phasen noch keine Benutzertests sondern Interviews und Diskussionen zu Konzepten und Ideen durchgeführt worden waren. Bei den Anwendungspartnern sollten der aktuelle Stand der drei Prototypen Waves IS, KISSy und Wiquila getestet werden – analog zur Vorgehensweise der beiden vorherigen Tests. Den Anwendern war WavesIS bereits aus früheren Releases bekannt. Die Prototypen KISSy und Wiquila wurden zum ersten Mal getestet.

Die Evaluation der Prototypen gliederte sich in zwei Stufen. In der ersten Stufe sollten die Werkzeuge vor Ort getestet und diskutiert werden. Hierfür wurden jeweils an unterschiedlichen Tagen Usability Workshops bei den Unternehmen Disy, PTV, Object International und CAS durchgeführt. Zu Beginn wurde der aktuelle Stand des Projektes vorgestellt sowie die Ziele und die Vorgehensweise präsentiert. Anschließend testeten die Mitarbeiter des Unternehmens die Prototypen im Rahmen von Einzelinterviews bzw. Usability Tests. In der zweiten Stufe sollten die Prototypen über eine Laufzeit von zwölf Wochen im laufenden Betrieb genutzt und im Rahmen von wöchentlichen Telefon-Interviews evaluiert werden. Hierbei wurden Nutzungsstatistiken erhoben und sowohl ein Projekttreffen als auch ein Abschlussworkshop veranstaltet.

Im Rahmen der Evaluation wurde für diese Phase des Forschungsprojektes die Möglichkeit geschaffen, OpenProposal einzusetzen und zu evaluieren. In Zusammenarbeit mit der Moderatorin wurde im Juli 2008 die Vorgehensweise in der Evaluation angepasst und die Fallstudie „WAVES“ mit einer Laufzeit von Juli bis Oktober 2008 konzipiert.

Die Rolle der Moderatorin nahm eine Mitarbeiterin der Arbeitsgruppe Software Engineering am Institut für Informatik der FU Berlin ein. Sie war für die Arbeitspakete Anforderungsanalyse und Evaluation im Projekt WAVES verantwortlich und somit auch für die Durchführung der Usability Workshops zuständig. Die WAVES Prototypen wurden von drei Teams, jedes bestehend aus zwei wissenschaftlichen Mitarbeitern (Entscheider) und zwei studentischen Entwicklern (Entwickler) am FZI entwickelt. Die Mitarbeiter der Anwendungsunternehmen - in der Mehrheit Software-Entwickler – nahmen die Rolle der Anwender ein.

9.5.2 Gestaltung der Fallstudie „WAVES“

Die Planung und Durchführung der Fallstudie erfolgte in enger Zusammenarbeit mit der Moderatorin. Die Gestaltung der Fallstudie umfasste dabei die Beschreibung der Ziele, die Auswahl der Teilnehmer, die Installation des technischen Aufbaus und die Planung des zeitlichen Ablaufs der Fallstudie.

9.5.2.1 Ziele der Fallstudie „WAVES“

Aus Sicht des Evaluationspartners FU Berlin war man an der Untersuchung der Einsatzpotentiale von OpenProposal interessiert und erhoffte sich dadurch eine Steigerung von konstruktiven Rückmeldungen der Anwendungspartner. Als Vergleichssystem zu OpenProposal dienten die anderen im Projekt eingesetzten Werkzeuge JIRA, Projekt-Wiki und Videoanalyse. Dabei sollte die geplante Vorgehensweise beibehalten und OpenProposal zusätzlich sowohl im Rahmen der Einzelinterviews als auch in der zwölf-wöchigen Testphase eingesetzt werden.

Im Rahmen der Fallstudie sollten dabei die Hypothesen H1 - H5 (vgl. Kapitel 9.3.2) untersucht werden. Es wurde der Fragenstellung nachgegangen, ob OpenProposal dazu beitragen kann, die Ergebnisse von Anwenderbefragungen zu verbessern und Anwender bei der Erstellung von Beiträgen im Rahmen ihrer täglichen Arbeit zu unterstützen. Als unabhängige Variablen UV1 wurde zwischen „UV1-1 OpenProposal“, „UV1-2 JIRA“, „UV1-3 Projekt-Wiki“, „UV1-4 Videoanalyse“ und „UV1-5 Feedbackformular“ variiert.

9.5.2.2 Teilnehmer der Fallstudie „WAVES“

An der Fallstudie nahmen 16 Anwender (zwei weiblich, 14 männlich), eine Moderatorin, drei Entscheider und drei Entwickler teil. Der Altersdurchschnitt der Anwender lag bei 35,36 Jahren (MW: 35,36; MIN: 29,00; MAX: 54,00; STABW: 5,92). Sie stammten aus den Entwicklungsabteilungen der Unternehmen (drei bei Object International, fünf bei PTV, vier bei CAS, vier bei disy) und wiesen unterschiedliche Erfahrungen im Umgang mit der Formulierung von Anforderungen auf; neun schätzten sich als erfahren und sieben als unerfahren ein.

Alle Anwender gaben an, dass sie regelmäßig Wikis und Suchmaschinen benutzen. WavesIS war allen Anwendern zum Zeitpunkt der Fallstudie bekannt, nicht jedoch KISSy, Wiquila oder OpenProposal. JIRA wurde bereits im Rahmen des Projektes und auch bei den Unternehmen disy und PTV für ihr eigenes Aufgabenmanagement eingesetzt.

Die Kommunikation mit den Anwendern lag im Aufgabenbereich der Moderatorin. Sie war für die Organisation und Auswertung der Usability Workshops und den Test im laufenden Betrieb verantwortlich.

Für jeden Prototyp waren jeweils ein Entscheider und ein Entwickler zuständig. Die Entscheider waren jeweils für ihre Prototypen verantwortlich und hatten hierfür volle Entscheidungskompetenz. Sie koordinierten die Aufgaben ihrer Entwickler und führten die Kommunikation mit der Moderatorin durch. Darüber hinaus war eine enge Zusammenarbeit zwischen den Entscheidern untereinander erforderlich, da ihre Prototypen auf einer gemeinsamen technischen Plattform aufbauten. Hierfür fanden regelmäßig Besprechungen mit allen Entscheidern und Entwicklern statt.

Zwei Forscher des Entwicklungsteams von OpenProposal waren bei den Unternehmen vor Ort dabei und standen für technische Fragen zu OpenProposal zur Verfügung. Bei den Einzelinterviews waren sie nicht anwesend. Die Auswertung der Einzelinterviews erfolgte über Videoaufnahmen.

9.5.2.3 Technischer Aufbau der Fallstudie „WAVES“

Zur Erstellung von Anwenderbeiträgen und zur Kommunikation mit der Moderatorin und den Entscheidern standen den Anwendern im Laufe der Studie mehrere Möglichkeiten zur Verfügung:

- Für die Einzelinterviews wurde bei jeden Unternehmen ein Testcomputer in einem Besprechungsraum bereitgestellt, an den sich ein Anwender und die Moderatorin ungestört setzen und die Prototypen testen konnten. Der Testcomputer diente auch zur Abgabe von Verbesserungsvorschlägen. Hierzu wurden auf dem Testcomputer die Software Camtasia Studio 5 und eine Videokamera als Videoaufnahmesystem installiert (vgl. UV1-4). Außerdem wurde OpenProposal auf den Testcomputer kopiert und für den Test eingerichtet (vgl. UV1-1). Da kein Netzwerkzugang an dem Testcomputer vorhanden war, wurde in OpenProposal der Sendemodus „lokal“ eingestellt, so dass die Vorschläge als Dateien lokal auf der Festplatte

gespeichert wurden. Nach den Einzelinterviews kopierte die Moderatorin die Anwenderbeiträge auf einen USB-Stick und übertrug diese anschließend mit einem an das Internet angeschlossenen Computer in JIRA. Zur Übertragung der Anwenderbeiträge in JIRA setzte sie OpenProposal ein. Hierbei stellte sie bei OpenProposal als Sendemodus „JIRA“ ein, öffnete jeden einzelnen Anwenderbeitrag und sendete diesen ab.

- Während des zwölf-wöchigen Tests im Betriebsalltag konnten die Anwender ihre Anwenderbeiträge mit OpenProposal erstellen und versenden. Als Sendemodus war „JIRA“ eingestellt, da die Testcomputer der Anwender mit einer Netzwerkverbindung ausgestattet waren. So wurden die Anwenderbeiträge beim Absenden direkt in JIRA übertragen und der Umweg über einen USB-Stick vermieden. Alternativ konnten die Anwender ihre Anwenderbeiträge während der Testphase in ein in WavesIS integriertes, webbasiertes Feedbackformular eingeben (vgl. UV1-5), einen Beitrag in JIRA eintragen (vgl. UV1-2) oder ihre Ideen im Projekt-Wiki notieren (vgl. UV1-3).

OpenProposal lag zum Zeitpunkt der Fallstudie in der Version 2.43 vor. Im Vergleich zur Version 2.0 aus der ersten Fallstudie wurden die Bedienoberfläche und die Symbole der Annotationswerkzeuge optisch überarbeitet.

Vor Beginn der Fallstudie wurde eine Schnittstelle eingerichtet, so dass die mit OpenProposal erstellten Vorschläge beim Absenden automatisch als Issue in JIRA angelegt und das annotierte Bildschirmfoto und die XML-Datei des Vorschlages als angehängte Dateien zugeordnet werden konnten. Die JIRA-Schnittstelle, die bereits in der Fallstudie „TRUMPF“ zum Einsatz gekommen war, konnte hierfür wiederverwendet werden.

Zudem kam eine Regeldatenbank zum Einsatz: In JIRA wurde für jeden der drei Prototypen WavesIS, KISSy und Wiquila ein eigenes Projekt geführt. In der Konfiguration von OpenProposal wurden in der Regeldatenbank mehrere Regeln hinterlegt, so dass die Vorschläge abhängig von der annotierten Anwendung einem entsprechenden Projekt zugeordnet wurden. Wurde beispielsweise ein Anwenderbeitrag zu WavesIS erstellt, erkannte OpenProposal über die Systemparameter den Namen der annotierten Anwendung und ordnete den Anwenderbeitrag beim Absenden dem in JIRA befindlichen Projekt „WavesIS“ zu.

9.5.2.4 Zeitlicher Ablauf der Fallstudie „WAVES“

Die Vorbereitung der Fallstudie begann Anfang 2008. Der Moderatorin wurde per Email eine Testversion von OpenProposal zum Ausprobieren gesendet. Sie fand OpenProposal interessant und war einverstanden, OpenProposal in der nächsten Evaluationsphase einzusetzen. Die technische Anbindung an den Projekt-Issue-Tracker JIRA wurde anschließend innerhalb weniger Minuten eingerichtet, da die gleiche Schnittstelle wie bei TRUMPF verwendet werden konnte.

Die Usability Workshops fanden am 04.07.2008 mit drei Teilnehmern bei Object International, am 08. 07.2008 mit vier Teilnehmern bei Disy, am 16.07.2008 mit fünf Teilnehmern bei PTV und am 06.08.2008 mit vier Teilnehmern bei der CAS statt. Bei jedem Unternehmen wurde die identische Vorgehensweise gewählt.

Zu Beginn wurde allen Anwendern in einer gemeinsamen Runde die Vorgehensweise, die zu testenden Prototypen und die Funktionsweise von OpenProposal vorgestellt. Anschließend wurden die

Anwender nacheinander in Einzelinterviews gebeten, die Prototypen mit vorgegebenen Aufgaben zu testen und Anwenderbeiträge abzugeben. Außerdem wurde OpenProposal allen Teilnehmern kurz vorgeführt. Es konnten Fragen zu den Prototypen und zu OpenProposal gestellt werden. Anschließend wurde an einem Testrechner in einem geschlossenen Raum mit den Einzeltests begonnen. Jeder Teilnehmer sollte ca. 30 bis 45 Minuten befragt werden. Während der Befragung wurden die Prototypen ausprobiert und anhand der „Thinking aloud“ Methode kommentiert. Die Moderatorin wies dabei vor und während des Testes auch darauf hin, dass die Vorschläge sowohl mündlich auf Video aufgezeichnet werden als auch schriftlich mit OpenProposal beschrieben werden konnten. Bei Bedarf konnten die Testpersonen somit OpenProposal einsetzen und ihre Ideen lokal auf der Festplatte des Testrechners speichern. Der Desktop und die Testperson wurden mit einer auf den Tester gerichteten Videokamera und einer Desktopaufzeichnungssoftware (Camtasia Studio 5) aufgezeichnet. Darüber hinaus machte sich die Moderatorin handschriftlich Notizen.

Nach jedem Test wurde jedem Anwender ein Fragebogen zu OpenProposal (ähnlich zu dem aus der ersten Fallstudie) ausgeteilt, der freiwillig ausgefüllt und bei der Moderatorin abgegeben werden konnte. Die mit OpenProposal erstellten Verbesserungsvorschläge wurden von der Moderatorin in JIRA übertragen und zur Bearbeitung freigegeben. Zum Schluss wurden die gesammelten Erkenntnisse in einer gemeinsamen Runde mit allen Anwendern diskutiert. Wenn möglich wurden die erfassten Anwenderbeiträge in einer gemeinsamen Diskussionsrunde zusammengefasst vorgestellt und diskutiert. Hierfür plante die Moderatorin zwischen dem letzten Einzelinterview und dem Beginn der Diskussionsrunde eine Stunde Zeit ein, um die Vorschläge aus OpenProposal und ihren Notizen zu sichten und eine kurze Präsentation vorzubereiten. Die Präsentation wurde dann über einen Videoprojektor angezeigt und die Diskussion anhand der Präsentation geleitet. Die Ergebnisse der Diskussion wurden direkt in die Folien der Präsentation eingegeben. Außerdem notierte sich die Moderatorin besonders wichtige Sachverhalte.

In den anschließenden zwölf Wochen wurden die Anwender gebeten, die beiden Prototypen für ihre tägliche Arbeit zu verwenden und weitere Anwenderbeiträge während der Nutzung wahlweise per Email, Telefon, JIRA, OpenProposal oder das in WavesIS integrierte Feedback-Formular abzugeben. Dabei hatten sie auch die Möglichkeit, sich die bereits abgegebenen Anwenderbeiträge in JIRA anzuschauen und den Bearbeitungsstatus ihrer eigenen Vorschläge zu verfolgen. Zudem nahm die Moderatorin alle zwei Wochen telefonisch mit jeweils einem dedizierten Ansprechpartner der vier Unternehmen Kontakt auf und befragte diese zu ihren Erfahrungen mit den Prototypen. Außerdem konnten die Anwender jederzeit telefonisch und per Email Kontakt mit der Moderatorin und den Entscheidern aufnehmen.

Am 20.10.2008 wurden die Ergebnisse des Projektes einschließlich der Ergebnisse der Evaluation der Prototypen in einem Abschlussworkshop mit allen Partnern präsentiert. Zu diesem Zeitpunkt endete auch das Projekt. In Form von narrativen Interviews wurden abschließend die Moderatorin, die Entscheider und die Entwickler zu ihren Erfahrungen mit OpenProposal befragt.

Zur Datenerhebung wurden die Einzelinterviews auf Video protokolliert, Fragebögen an die Anwender ausgeteilt, die erstellten Anwenderbeiträge ausgewertet und Befragungen mit der Moderatorin, dem Entscheider und einem Entwickler durchgeführt. Abschließend wurde ein Fallstudienbericht und eine Präsentation der Ergebnisse erstellt, die mit der Moderatorin abgestimmt und zur Durchsicht an alle Teilnehmer versendet wurde.

9.5.3 Ergebnisse der Fallstudie „Waves“

Der Verlauf der Studie erfolgte gemäß dem zeitlichen Ablaufplan. Dabei konnte eine Vielzahl von Daten erhoben werden, anhand derer die Hypothesen zu OpenProposal überprüft wurden. Im Folgenden werden die Ergebnisse der Datenerhebung in Form von Beobachtungsprotokollen sowie anhand der erstellten Anwenderbeiträge, der ausgefüllten Fragebögen und der Experteninterviews zusammengefasst aufgeführt.

9.5.3.1 Auswertung der Beobachtungsprotokolle

Als Beobachtungsprotokolle lagen eigen angefertigte Notizen zu den Beobachtungen der Vor- und Nachbereitung der Usability Workshops und zur Kommunikation mit der Moderatorin und den Entscheidern sowie die Videoaufzeichnungen der Einzelinterviews vor. Im Vorfeld der Einzelinterviews und im Anschluss daran wurden die Beobachtungen des Verlaufs handschriftlich notiert. Der Ablauf der Einzelinterviews wurde den Videoaufnahmen der Moderatorin entnommen. Während des Tests im laufenden Betrieb standen die Anwender nicht unter Beobachtung, da die Anwender ihrer normalen Arbeit nachgingen und eine Beobachtung ethisch nicht vertretbar gewesen wäre. Stattdessen wurden die Anwender, die Moderatorin und die Entscheider gebeten, die Forscher bei auffälligen Ereignissen zu informieren.

Hauptaugenmerk der Beobachtungsprotokolle lag auf der Datenerhebung der Variablen AV7 „Aufwand des Moderators für die Vorbereitung zur Erfassung von Anwenderbeiträgen“, AV8 „Aufwand des Moderators für die Nachbereitung der erstellen Anwenderbeiträge“, AV9 „Aufwand des Moderators für die Interaktion mit den Anwendern“ und AV6 „Zufriedenheit der Anwender mit der Methode zur Anwenderbeteiligung“. Somit wurde zum einen der Aufwand zur Vor- und Nachbereitung und Durchführung der Anwenderbeteiligung erfasst und zum anderen die Zufriedenheit der Anwender untersucht.

Die Einzelinterviews bei Object International liefen nicht gemäß den Erwartungen: Keiner der drei Anwender benutzte OpenProposal in den Einzelinterviews. In der anschließenden gemeinsamen Diskussionsrunde wurden hierfür zwei Gründe genannt. Erstens konnten die Anwender während des Interviews ihre Anwenderbeiträge mündlich abgeben, so dass dies aus ihrer Sicht ausreichend und eine erneute Beschreibung mit OpenProposal nicht nötig war. Zudem wurde angegeben, dass die Prototypen aus ihrer Sicht in einem frühen Zustand seien und konkrete Verbesserungsvorschläge zu diesem Zeitpunkt nicht angefallen wären. Nach ausführlicherem Nachfragen zeichnete es sich ab, dass vor allem der erste Grund ausschlaggebend war. Daher wurde von den Forschern vorgeschlagen, dass die Moderatorin ihre Vorgehensweise bei den nachfolgenden Einzelinterviews bei den anderen Unternehmen anpasste: Sie sollte die Anwender im Test darauf hinweisen, ihre Anwenderbeiträge sowohl mündlich als auch mit OpenProposal abzugeben. Dieser Vorschlag wurde von der Moderatorin akzeptiert und umgesetzt.

Bei den weiteren Einzelinterviews bzw. Usability Tests in den anderen drei Unternehmen waren die Anwender zu Beginn ähnlich wie bei Object International irritiert, dass sie ihre Anwenderbeiträge gleichzeitig mündlich und mit OpenProposal beschreiben sollten. Dieser Irritation konnte die Moderation mit der angepassten Vorgehensweise entgegenwirken und die Anwender zur Abgabe ihrer Verbesserungsvorschläge sowohl laut aussprechend als auch mit OpenProposal motivieren.

Der Umgang mit OpenProposal wurde unterschiedlich gehandhabt. Bei acht Anwendern blieb das Fenster von OpenProposal ständig im Vordergrund geöffnet. Vier minimierten das Fenster und riefen es erst bei Bedarf auf. Ein Anwender beendete OpenProposal nach jedem Absenden eines Beitrags. Alle Anwender kamen mit der Bedienung von OpenProposal schnell zurecht. Viele der Anwender bemühten sich, das passende Annotationswerkzeug für ihren Beitrag zu suchen. Insgesamt wurden mit OpenProposal 75 Anwenderbeiträge abgegeben. Hinzu kamen 43 Anwenderbeiträge, die mündlich kommuniziert wurden. Bei 22 der mündlich beschriebenen Anwenderbeiträge bat die Moderatorin die Anwender darum, ihre Ideen zu konkretisieren. Hierzu griffen die Anwender in 20 Fällen auf OpenProposal zurück. In den anderen zwei Fällen konkretisierten sie ihren Vorschlag mündlich. Beispielhaft war folgender Dialog zwischen Moderatorin und Anwender:

> *Anwender: „Die Darstellung der Suchergebnisse ist etwas unübersichtlich."*
>
> *Moderatorin: „Wie könnte man das verbessern?"*
>
> *Anwender: „Beispielsweise könnte man die Ergebnisse nach den Datenquellen sortieren."*
>
> *Moderatorin: „Wie genau könnte das aussehen? Können Sie das mit OpenProposal verdeutlichen?"*

Im Anschluss an die Einzelinterviews fanden bei Object International und Disy gemeinsame Diskussionsrunden mit der Moderatorin und den Anwendern statt. Aufgrund von zeitlichen Engpässen der Anwenderunternehmen musste bei PTV und CAS auf die gemeinsame Diskussionsrunde verzichtet werden. Hierfür fasste die Moderatorin die Hauptergebnisse der Einzelinterviews anhand ihrer Notizen auf Präsentationsfolien zusammen und stellte diese den Anwendern in der gemeinsamen Runde vor. Die Moderatorin notierte sich den Diskussionsverlauf dabei handschriftlich. Die Anmerkungen der Anwender waren dabei vorwiegend allgemeiner Natur. Einer der Anwender äußerte sich beispielsweise folgendermaßen:

> *„Wir sind häufig beim Kunden vor Ort und arbeiten mit anderen Teams zusammen. In der Code-Suchmaschine sollten daher auch die Datenquellen der anderen Teams integriert werden können. Außerdem sollte auch C++-Quellcode unterstützt werden, damit das überhaupt Sinn für uns macht."*

Für die gemeinsame Diskussionsrunde bei Disy integrierte die Moderatorin die mit OpenProposal erstellten Anwenderbeiträge in ihre Präsentation. Die Diskussionen erfolgten ähnlich wie bei Object International, außer dass bei den OpenProposal Anwenderbeiträgen auch die konkrete Ausprägung der Beiträge diskutiert wurde.

Im Anschluss an die Vor-Ort-Termine bei den Unternehmen wertete die Moderatorin die Anwenderbeiträge aus und kommunizierte diese an die Entscheider. Die Nachbereitung der OpenProposal Vorschläge erfolgte direkt nach den Einzelinterviews und benötigte wenige Minuten, um identische Vorschläge auszusortieren und ungünstige Titelbeschreibungen anzupassen. Den Entscheidern standen die OpenProposal Vorschläge somit direkt nach den Einzelinterviews in JIRA zur Verfügung. Die Auswertung der Videoaufnahmen benötigte nach Auskunft der Moderatorin ca. zwei Stunden pro Interview. Die Ergebnisse konnten somit erst mehrere Wochen nach den Interviews über Telefon, JIRA oder Projekt-Wiki kommuniziert werden. Die wichtigsten Anmerkungen wurden telefonisch von der Moderatorin und den Entscheidern besprochen. Alle Vorschläge, die sich direkt umsetzen ließen,

übertrug sie in JIRA und ordnete sie den Entscheidern zu. Im Projekt-Wiki wurden diejenigen Vorschläge abgelegt, die allgemeiner Natur waren und im Anforderungsdokument des Projektes ergänzt wurden.

Im anschließenden laufenden Betrieb griffen die Teilnehmer acht Mal auf die Möglichkeit zur Abgabe von Vorschlägen zurück. Auf Rückfrage der Moderatorin begründeten die Anwender die niedrige Anzahl der erstellen Anwenderbeiträge mit ihrer knappen Zeit während ihrer täglichen Arbeit. Nur bei schwerwiegenden Fehlern entstand Handlungsbedarf. In diesen Fällen wurde OpenProposal verwendet und als hilfreich empfunden. Dabei wurde das Problem zunächst telefonisch an die Moderatorin bzw. an die Entscheider gemeldet und anschließend die Beschreibung mit OpenProposal erstellt. JIRA, das Projekt-Wiki und das integrierte Feedbacksystem wurden kein einziges Mal verwendet.

Während der Interviews und Diskussionen wurden die Vor- und Nachteile von OpenProposal von den Anwendern und der Moderatorin angesprochen. Insgesamt fanden die Anwender den Ansatz von OpenProposal hilfreich und geeignet für eine gute Ergänzung zu den üblichen Methoden. Beispielsweise merkten die Anwender an:

> *„Bessere Visualisierung als nur textliche oder sprachliche Umschreibungen. Man sieht schneller und präziser, was sich der Kunde wünscht.“*

> *„Durch Visualisierung sind Vorschläge besser zu verstehen und leichter umzusetzen. Ein Bild sagt mehr als 1000 Worte."*

Dabei stellten die Anwender auch Erwartungen an der Verbesserung der Verständlichkeit der Anwenderbeiträge, die mit OpenProposal erstellt werden. Ein Anwender stellte fest:

> *„1. Weil die Vorschläge verständlicher und besser strukturiert ankommen. 2. Weil die Anforderungen auch dokumentiert sind und nicht unter den Tisch fallen.“*

Dabei wurden auch Verbesserungsvorschläge zu OpenProposal abgegeben. Es wurden unter anderem die Implementierung von einem Tastaturkürzel zum Aktivieren von OpenProposal und von neuen Formen von Annotationswerkzeugen angefragt:

> *„Umschalten zum Programm etwas umständlich, Tasten Kombination wäre praktisch.“*

> *„neben "Button einfügen" wäre "Icon einfügen" (Symbolleiste) praktisch; wäre cool wenn man Bereiche des bestehenden Screens verschieben könnte, oder geht das schon?“*

Die Moderatorin sah in OpenProposal eine Methode, mit der Anwender zur Konkretisierung ihrer Vorschläge motiviert werden. Die Anwender mussten allerdings darauf hingewiesen werden, neben der mündlichen Mitteilung auch OpenProposal zu verwenden.

Bezüglich des Aufwandes der Vor- und Nachbereitung war zur Installation von OpenProposal lediglich die Anbindung an JIRA notwendig. Für die Vorbereitung der Befragung entstand weder mit der Videoanalyse noch mit OpenProposal ein zusätzlicher Aufwand. Auch die Nachbereitung der Anwenderbeiträge in JIRA erforderte keinen zusätzlichen Aufwand. Im Gegensatz zu OpenProposal erforderte die Nachbereitung der Videoanalyse jedoch einen großen zeitlichen Aufwand für die Moderatorin.

Die Benutzung des Projekt-Wikis benötigte lediglich einen geringen Aufwand zur Eingabe der Anforderungsbeschreibungen.

9.5.3.2 Auswertung der erstellten Anwenderbeiträge

Die mit OpenProposal erstellen Anwenderbeiträge lagen nach Abschluss der Einzelinterviews und dem laufenden Betrieb in JIRA vor. Außerdem stellte die Moderatorin ihre Videoaufnahmen zur Auswertung zur Verfügung. In Bezug auf die abhängigen Variablen AV1 „Anzahl der erstellten Anwenderbeiträge“, AV2 „Umfang der erstellten Anwenderbeiträge“, AV3 „Konstruktivität der erstellten Anwenderbeiträge“, AV10 „Verständlichkeit der erstellten Anwenderbeiträge“ wurden bei der Auswertung die Anzahl, der Umfang, die Konstruktivität und die Verständlichkeit der erstellten Verbesserungsvorschläge ermittelt und darüber hinaus auf Auffälligkeiten geachtet.

In den Einzelinterviews wurden mit OpenProposal insgesamt 75 Anwenderbeiträge erstellt. Außerdem wurden 23 Anwenderbeiträge ausschließlich aus den Videoaufnahmen abgeleitet. Während des Tests im laufenden Betrieb wurden zusätzlich fünf Anwenderbeiträge mit OpenProposal erstellt. Mit den anderen Werkzeugen wurden keine weiteren Anwenderbeiträge erstellt. Eine Übersicht über die erstellten Anwenderbeiträge und die Verteilung der abgegebenen Anwenderbeiträge auf die Anwender stellt Abbildung 9.7 dar.

Die in den Einzelinterviews mit OpenProposal erstellten Anwenderbeiträge enthielten durchschnittlich eine Annotation mit einer aus 12,74 Wörter (MW: 12,75; Median: 10,00; MIN: 1,00; MAX: 55,00; STABW: 7,43) bestehenden erläuternden Beschreibung sowie einen Titel und eine allgemeine Kurzbeschreibung mit insgesamt 1,83 Wörtern (MW: 1,83; Median: 2,00; MIN: 0,00; MAX: 6,00; STABW: 0,79). Insgesamt wurden bei jedem OpenProposal Vorschlag durchschnittlich 14,57 Wörter (MW: 14,57; Median: 12,00; MIN: 1,00; MAX: 57,00; STABW: 7,75) eingegeben.

Als Annotationswerkzeug wurde in 30 Fällen das „Kommentar“-Werkzeug, in vier Fällen das „Verschieben“-Werkzeug und in 41 Fällen das „Hinzufügen“-Werkzeug verwendet. Die anderen Annotationswerkzeuge wurden nicht verwendet. Insgesamt waren 70 der 75 OpenProposal-Vorschläge konstruktive Verbesserungsideen. Die restlichen fünf Vorschläge waren Problemmeldungen bzw. Fragen. In einer nachträglichen Betrachtung gaben die Entscheider an, dass von den 30 Annotationen, die als „Kommentar“ kategorisiert wurden, in 26 Fällen eine andere Kategorisierung besser geeignet gewesen wäre. Davon hätten 13 Vorschläge als „Verbesserung“, acht als „Hinzufügen“ und fünf als „Frage“ kategorisiert werden können.

Von den acht mit OpenProposal im laufenden Betrieb erstellten Vorschlägen waren fünf Problemmeldungen und drei konstruktive Verbesserungsvorschläge der Kategorie „Verbessern“. Die Vorschläge enthielten durchschnittlich eine Annotation mit einer aus 1,80 Wörter (MW: 1,80; Median: 2,00; MIN: 0,00; MAX: 5,00; STABW: 1,44) bestehenden erläuternden Beschreibung. Der Titel bestand aus durchschnittlich 1,20 Wörtern (MW: 1,20; Median: 1,00; MIN: 0,00; MAX: 3,00; STABW: 1,04). Eine allgemeine Beschreibung wurde bei keinem der Vorschläge eingegeben. Insgesamt wurden bei den OpenProposal Vorschlägen durchschnittlich 3,00 Wörter (MW: 3,00; Median: 3,00; MIN: 0,00; MAX: 9,00; STABW: 2,40) eingegeben.

Bei der Sichtung der Anwenderbeiträge führten zwei Entscheider und zwei Entwickler eine Bewertung der eingegangen Anwenderbeiträge durch. Als Bewertungsskala diente die Schulnotenskale von 1,00 für sehr gut bis 6,00 für ungenügend. Die mit OpenProposal abgegebenen Anwenderbeiträge

wurden durchschnittlich mit der Note 1,67 (Mittelwert: 1,34; Median 2,00; MIN: 1,00; MAX: 4,00; STABW: 0,66) bewertet. Es mussten nachträglich 8,0% der OpenProposal Beiträge überarbeitet werden. Insgesamt wurde die Formulierung der Titel häufig als ungeeignet beurteilt, da diese in vielen Fällen sehr allgemein und knapp formuliert wurden. Aus ihrer Sicht vergrößert der Titel jedoch die Übersicht bei der Bearbeitung einer großen Liste von Anwenderbeiträgen und spielt daher eine wichtige Rolle. Bei der Diskussion über eine mögliche automatisierte Generierung der Titel und Beschreibung waren alle der Meinung, dass dies eine sinnvolle Lösung wäre. Dabei sollte der Titel aus der annotierten Anwendung, der Kategorie der Annotation und dem ersten Satz bzw. der ersten sechs Wörter der Annotation bestehen. Die Beschreibung sollte den Namen der annotierten Anwendung, die Annotation mit seiner Kategorie und die Systemparameter enthalten.

Mit OpenProposal hatten alle Anwender - bis auf die drei Anwender von Object International - einen oder mehrere Anwenderbeiträge erstellt. Im laufenden Betrieb wurde OpenProposal von vier unterschiedlichen Anwendern aus drei unterschiedlichen Unternehmen eingesetzt. Mit der Videoanalyse kamen alle Anwender zurecht.

Aus den Videos fasste die Moderatorin insgesamt 13 Anwenderbeiträge zusammen. Drei der 13 Anwenderbeiträge waren konstruktive Verbesserungsideen. Bei den zehn anderen Anwenderbeiträgen handelte es sich um reine Problemmeldungen. Vier Anwenderbeiträge notierte sie ins Projekt-Wiki. Neun Anwenderbeiträge wurden in nachfolgenden Projekttreffen im Oktober 2008 auf Präsentationsfolien präsentiert und in der Runde mit den Entscheidern diskutiert. In JIRA wurde von ihr kein Anwenderbeitrag eingetragen. Die Bestimmung der Wörteranzahl der Anwenderbeiträge aus der Videoanalyse war nicht möglich, da die Vorschläge nicht direkt von den Anwendern sondern von der Zusammenfassung der aufgezeichneten Videos der Moderatorin stammen.

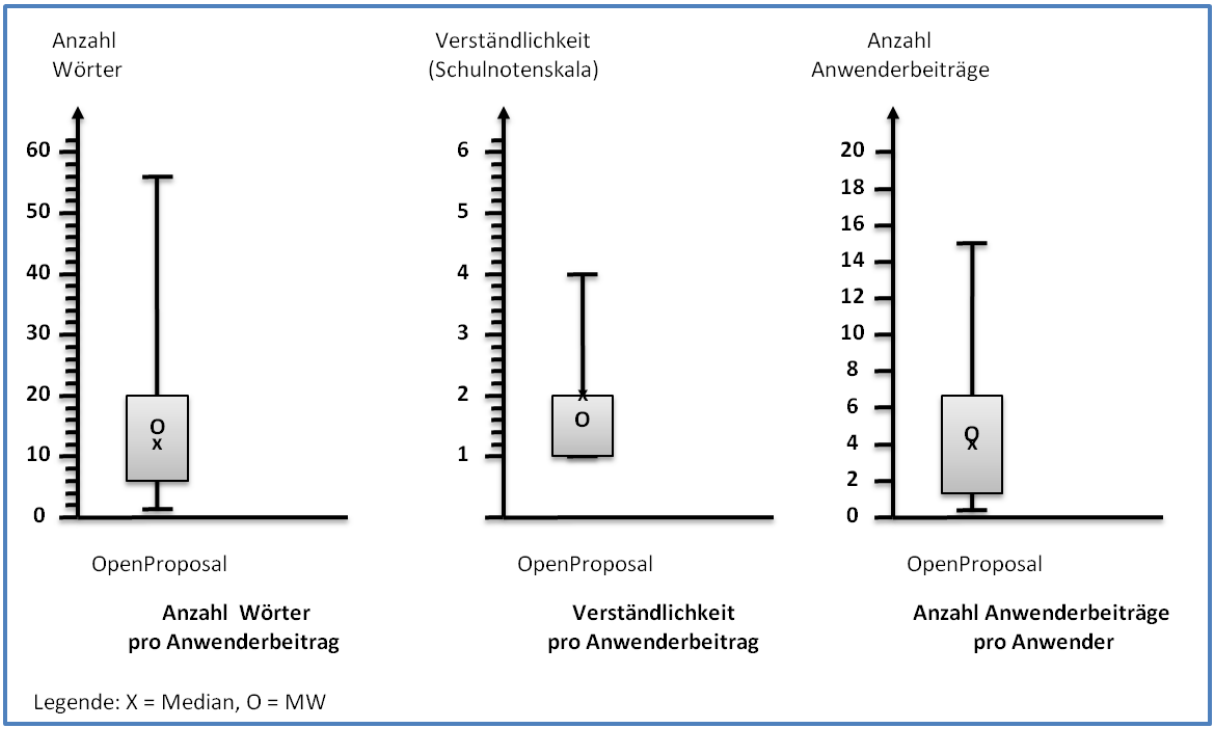

Abbildung 9.7: Übersicht über die abgegebenen Vorschläge in der Fallstudie „WAVES“

Auf das Projekt-Wiki, das Feedbackformular und JIRA hatten die Anwender während der Interviews keinen Zugriff. Im laufenden Betrieb wurden sie nicht verwendet.

Zusammenfassend können aus der Analyse der erstellten Anwenderbeiträge folgende Erkenntnisse gewonnen werden:

- Mit OpenProposal wurden sowohl in den Einzelinterviews als auch im laufenden Betrieb mehr Anwenderbeiträge als mit den anderen Methoden abgegeben. Allerdings wurde OpenProposal im laufenden Betrieb nur in wenigen Fällen verwendet. Mit anderen Methoden wurde im laufenden Betrieb kein Vorschlag abgegeben. Die Kategorisierung der Anwenderbeiträge wurde nur bei OpenProposal angewendet. Allerdings wurden nicht das gesamte Spektrum und häufig auch nicht die richtige Kategorie ausgewählt.

- Die Verständlichkeit der OpenProposal Anwenderbeiträge wurde von den Entscheidern und den Entwicklern als sehr gut, die Formulierung der Titel jedoch als schlecht bewertet und es wurde die Notwendigkeit einer automatisierten Titelgenerierung aufgeworfen.

- Mit den Werkzeugen JIRA, Projekt-Wiki und Feedbackformular wurde kein Anwenderbeitrag erstellt.

- Die Videoanalyse mit der „Thinking aloud“-Methode wurde von allen Anwendern akzeptiert. Während der Interviews nutzen die Anwender die Möglichkeit, ihre Eindrücke und Meinungen mündlich auszusprechen und mit der Moderatorin zu diskutieren.

9.5.3.3 Auswertung der Fragebögen

Jedem Anwender wurde nach den Einzelinterviews ein Fragebogen ausgeteilt. Dieser Fragebogen war dem Fragebogen aus der Fallstudie 1 „TRUMPF“ sehr ähnlich. Allerdings wurden die vergleichenden Fragen zu JIRA herausgenommen, da JIRA in der zweiten Fallstudie nicht als direktes Vergleichssystem bei den Einzelinterviews eingesetzt wurde. Stattdessen wurden die Anwender gebeten, die ihnen bekannten Methoden mit OpenProposal zu vergleichen. Die Forscher waren beim Ausfüllen der Fragebögen nicht im Raum anwesend und waren darüber hinaus um eine geringstmögliche Beeinflussung der Anwender bemüht. Mithilfe der Fragebögen sollte die Benutzerakzeptanz von OpenProposal untersucht werden und somit Daten zu den abhängigen Variablen AV5 „Komplexität der Erstellung von Anwenderbeiträgen“ und AV6 „Zufriedenheit der Anwender mit der Methode zur Anwenderbeteiligung“ erhoben werden. Da auch allgemeine Kommentare abgegeben werden konnten, konnten auch möglicherweise Antworten zu anderen Variablen erfasst werden.

Der Fragebogen gliederte sich in fünf Teile. Mit dem Deckblatt wurde der Anwender über die Ziele und Aufbau des Fragebogens und über die Vorgehensweise zur Anonymisierung informiert. Außerdem wurden wichtige Hinweise zum Ausfüllen des Fragebogens gegeben, u.a. dass in der Fußzeile jeder Seite des Fragebogens zusätzliche allgemeine Kommentare zum Fragebogen abgegeben werden konnten. Abschließend wurde den Anwendern für ihre Bemühungen gedankt.

In zweiten Teil wurden einleitend allgemeine Angaben zur Person erfragt. Hierzu gehörten Fragen zur Berufserfahrung, zum Geschlecht, zur Tätigkeit bzw. Fachrichtung und zu testrelevanten Handicaps. Ferner wurde die Testperson nach ihren bisherigen Erfahrungen zur Formulierung von Anforderun-

gen befragt und um eine Selbsteinschätzung ihrer Fähigkeit zur Formulierung von Anforderungen gebeten. Außerdem wurden Fragen zur Vertrautheit mit der getesteten Software gestellt.

In den drei weiteren Teilen des Fragebogens wurde der Anwender zur Einschätzung der Gebrauchstauglichkeit und Nützlichkeit von OpenProposal befragt. Dabei konnte der Anwender seine Einschätzung zu vorformulierten Thesen auf einer Likert-Skala mit sieben Merkmalsausprägungen abgeben. Als Antwortmöglichkeiten standen zu jeder These die Ausprägungen „Trifft überhaupt nicht zu“, „Trifft größtenteils nicht zu“, „Trifft eher nicht zu“, „Weder noch“, „Trifft eher zu“, „ Trifft größtenteils zu“ und „Trifft voll und ganz zu“ zur Auswahl. Außerdem konnte der Anwender sich einer Antwort enthalten bzw. mit „Ist mir nicht aufgefallen“ antworten. Um der Akquieszenz, der Tendenz den vorgegebenen Thesen unabhängig von ihrem Inhalt zuzustimmen, vorzubeugen, wurden die Thesen für das jeweilige Werkzeug abwechselnd positiv und negativ formuliert.

Die Thesen im dritten Teil bezogen sich auf die Darstellung der grafischen Benutzeroberfläche, auf die Hilfestellungen im Programm, auf die Bedienbarkeit und auf den Reifegrad von OpenProposal. Im vierten Teil wurden Thesen zur Nützlichkeit von OpenProposal aufgelistet. Im fünften Teil wurden vergleichende Thesen zu OpenProposal und anderen dem Anwender bekannten Werkzeugen aufgestellt, um den Anwender OpenProposal mit den ihm vertrauten Werkzeugen direkt miteinander vergleichen zu lassen.

Von den 16 ausgeteilten Fragebögen wurden neun Fragebögen vollständig und auswertbar ausgefüllt. Da bei Object International keiner der Anwender OpenProposal verwendet hatte, verzichteten deren teilnehmende Anwender auf das Ausfüllen der Fragebögen. Vier weitere Fragebögen waren nicht vollständig ausgefüllt.

Tabelle 9.8 und Tabelle 9.9 listen die wichtigsten Thesen zu OpenProposal auf und fasst die Antworten mit dem ermittelten Mittelwert (MW), der Varianz (VAR) sowie Minimum (MIN) und Maximum (MAX) zusammen. Die Antworten sind folgendermaßen kodiert: „Enthaltungen = Ist mir nicht aufgefallen“, „1,00 = Trifft überhaupt nicht zu“, „2,00 = Trifft größtenteils nicht zu“, „3,00 = Trifft eher nicht zu“, „4,00 = Weder noch“, „5,00 = Trifft eher zu“, „6,00 = Trifft größtenteils zu“ und „7,00 = Trifft voll und ganz zu“.

Die Bewertungen der Anwender zu OpenProposal (vgl. Tabelle 9.8) fielen im Durchschnitt sehr positiv aus. Die Antworten weisen darauf hin, dass die Anwender mit der Bedienung von OpenProposal gut zurechtkamen und Vorschläge schnell und unkompliziert abgegeben werden konnten. Die Symbolik und Namensgebung schien im Vergleich zu den anderen Aspekten verbesserungswürdig zu sein. Die Varianz zwischen den Antworten von allen Anwendern war bis auf eine Ausnahme sehr niedrig. Bei dieser Ausnahme gaben zwei Anwender an, dass sie bei OpenProposal oft steckengeblieben waren. Aufgrund der Angaben in den freien Kommentarfeldern war dies darauf zurückzuführen, dass sie mit dem Aktivieren bzw. Deaktivieren von OpenProposal Schwierigkeiten hatten.

Zur Frage nach ihrer üblichen Vorgehensweise wurden der Einsatz von Bildschirmfotoprogrammen, Issue-Tracker, Grafikprogrammen und elektronischen Textdokumenten angegeben. Außerdem gab ein Anwender an, dass er Bildschirmfotos erstellt, ausdruckt und seine Vorschläge einzeichnet. Einer der Anwender erwähnte den Einsatz von UML-Diagrammen.

Thesen	MW	VAR	MIN	MAX
Ich habe mich schnell in OpenProposal zurecht gefunden	6,20	0,18	6,00	7,00
OpenProposal zeigt zu viel Information auf einmal; das hat mich verwirrt.	1,90	0,54	1,00	3,00
Symbole und Namensgebung in OpenProposal sind eingängig.	5,56	0,53	5,00	7,00
Das Erstellen von Vorschlägen war unnötig kompliziert und langwierig.	1,50	0,5	1,00	3,00
Vorschläge sind einfach und ohne viel Nachdenken zu müssen zu erstellen.	6,33	0,25	6,00	7,00
Ich bin bei OpenProposal oft stecken geblieben und musste einen anderen Weg suchen.	2,78	1,94	1,00	5,00

Tabelle 9.8: Antworten zu den Thesen zu OpenProposal im Fragebogen der Fallstudie „WAVES“

Im Vergleich in Tabelle 9.9 zwischen OpenProposal und der bisher üblichen Vorgehensweise der Anwender konnte ein positives Ergebnis für OpenProposal ermittelt werden. Insgesamt konnte festgestellt werden, dass die Anwender den weiteren Einsatz von OpenProposal begrüßen würden. Die bisher übliche Vorgehensweise ohne OpenProposal würden sie ablehnen. Außerdem wurde die Erstellung von Anwenderbeiträgen mit OpenProposal als einfacher und weniger aufwändig bewertet. Dabei spielte die Art der üblichen Vorgehensweise keine Rolle, da alle Anwender trotz unterschiedlicher Vorgehensweisen sehr ähnliche Bewertungen abgaben.

Thesen	MW	VAR	MIN	MAX
Ich würde den Ansatz von OpenProposal der sonst üblichen Vorgehensweise vorziehen.	5,56	1,28	4,00	7,00
Ich würde meine übliche Vorgehensweise eher einsetzen als OpenProposal.	2,78	1,44	1,00	4,00
Ich finde die Vorschlagserstellung bei OpenProposal einfacher als bei meiner üblichen Vorgehensweise.	5,67	1,5	4,00	7,00
Ich finde die Vorschlagserstellung ist bei OpenProposal mit weniger Aufwand verbunden als bei meiner üblichen Vorgehensweise.	5,67	1,25	4,00	7,00
Für mich käme nur eine Kombination von meiner üblichen Vorgehensweise und OpenProposal in Frage, wenn es um die Vorschlagserstellung geht.	3,88	2,98	1,00	6,00

Tabelle 9.9: Antworten zu den vergleichenden Thesen im Fragebogen der Fallstudie „WAVES“

Weniger eindeutig war allerdings die Aussage ob sie OpenProposal exklusiv oder nur in Kombination mit der bisher üblichen Vorgehensweise verwenden würden. Fünf Anwender würden OpenProposal ihrer üblichen Vorgehensweise vorziehen und keine Kombination für notwendig halten. Zwei Anwender würde eher eine Kombination von OpenProposal mit ihrer üblichen Vorgehensweise bevorzugen und antworten zur Frage, ob sie OpenProposal der üblichen Vorgehensweise vorziehen würden, mit „Weder noch“. Zwei Anwender würden sowohl den Einsatz von OpenProposal der üblichen Vorgehensweise vorziehen als auch eine Kombination von beiden für sinnvoll halten.

Acht Anwender stimmten darüber hinaus zu, dass durch das Annotieren von Bildschirmfotos mithilfe von OpenProposal die Verständlichkeit der Anwenderbeiträge erhöht und damit auch eine höhere Annahmequote ihrer Anwenderbeiträge erwartet werden kann:

> *„Man sieht auf einen Blick, was gemeint ist. Keine / weniger Fehlinterpretationen. Zeitersparnis“*

> *„Bessere Visualisierung als nur textliche/sprachliche Umschreibung; man sieht schneller und präziser, was sich der Kunde wünscht“*

Ein Anwender stimmte dieser Erwartung nicht zu. Er sah in OpenProposal zwar eine sinnvolle Methode zur Unterstützung der Anforderungserhebung aber erwartete keine Beeinflussung:

> *„Ist eher ein Werkzeug zur Unterstützung, als zum Ablösen klassischer Anforderungen.“*

Neben den Anwenderbeiträgen, die schon in den Gesprächen während den Tests geäußert wurden, gaben die Anwender in den Kommentarfeldern im Fragebogen weitere Anmerkungen ab. Als Vorteile wurde hervorgehoben, dass OpenProposal eine gute und schnelle Handhabung bietet und die Erstellung von Anwenderbeiträgen erleichtert:

> *„Intuitiv bedienbar, einfach bedienbar, eindeutige Dokumentation“*

> *„Gute und schnelle Handhabung“*

> *„Verständlicher, einfacher, gute Kommunikation von Kunden, professionell, motivierend“*

Ein Anwender war zudem daran interessiert, wie die Anwenderbeiträge weiter bearbeitet und ins Anforderungsmanagement integriert werden. Seine Anmerkung lautete:

> *„Inwieweit ist angedacht, die erstellten Anforderungen dann auch systematisch abarbeiten zu können? Integration in Anforderungsmanagement?“*

Er sah hier eine Notwendigkeit darin, dass die OpenProposal Anwenderbeiträge auch für den Moderator und Entscheider handhabbar bleiben und in den Entwicklungsprozess passen.

9.5.3.4 Auswertung der Experteninterviews

Nach Abschluss der Benutzertests und Ende des Forschungsprojektes WAVES wurden separate Experteninterviews mit der Moderatorin, den drei Entscheidern und den drei Entwicklern durchgeführt. Für die Experteninterviews wurde analog zur ersten Fallstudie ein Leitfaden erstellt, um den Befragungen eine grobe Struktur vorzugeben ohne dabei das Gespräch jedoch zu stark auf bestimmte Fragestellungen einzuschränken. Der Interview-Leitfaden fußte auf den Variablen AV3 „Konstruktivität der erstellten Anwenderbeiträge“, AV7 „Aufwand des Moderators für die Vorbereitung zur Erfassung von Anwenderbeiträgen“, AV8 „Aufwand des Moderators für die Nachbereitung der erstellten Anwenderbeiträge“, AV9 „Aufwand des Moderators für die Interaktion mit den Anwendern“ und AV10 „Verständlichkeit der erstellten Anwenderbeiträge“ auf und beinhalte somit Fragen zum Aufwand der Vor- und Nachbereitung der Workshops, der Verständlichkeit und Konstruktivität der Vorschläge. Jedes Gespräch wurde aufgezeichnet und ein Mitschrieb angefertigt, auf deren Basis die Auswertung angefertigt wurde.

Aus Sicht der Moderatorin ergaben sich mit OpenProposal zahlreiche Vorteile gegenüber den üblichen Werkzeugen. Den größten Vorteil sah sie darin, dass die Anwender mit OpenProposal gezwungen werden, sich festzulegen wie die Umsetzung ihrer Vorschläge aussehen soll. Dies war ihr vor allem im Vergleich zu den früheren Evaluationsphasen MR0 und MR1 aufgefallen. Weitere Vorteile sah sie im geringen Aufwand der Nachbereitung und der zeitnahen Kommunikation an die Entscheider. Den Anwendern schien OpenProposal ihrer Meinung nach eine Abwechslung zu den üblichen Werkzeugen darzustellen:

> *„Den Anwendern hat es Spaß gemacht. Einer sagte sogar: Das ist ja lustig. Es gab dabei nicht die gewohnten langen Gesichter.“*

> *„OpenProposal kam total positiv an. Die Benutzer waren auf OpenProposal neugierig.“*

Allerdings musste sie die Anwender dazu motivieren, OpenProposal einzusetzen und Beiträge schriftlich fixieren. Dies ist darauf zurückzuführen, dass die Anwender es bevorzugten, ihre Vorschläge mündlich zu kommunizieren. Dies zeigte sich auch darin, dass OpenProposal zeitaufwändiger als die mündliche Formulierung war und das Aufschreiben Zeit und Konzentration kostete. Außerdem bestand die Gefahr, dass sich die Anwender durch die Anwesenheit der Moderatorin unter Druck fühlen, wenn diese auf die Fortsetzung des Tests wartete. Diese gab jedoch an sich insgesamt eher zurückhaltend verhalten zu haben, um zu sehen, ob OpenProposal aus eigener Initiative genutzt würde.

Für Anwenderbeiträge zu grafischen Benutzeroberflächen wurde OpenProposal als gut geeignet eingeschätzt, nicht jedoch für Funktionalitäten, die nicht auf der Oberfläche zu finden sind, z.B. „Ranking der Suche“ oder „Dateiname ist immer wichtiger als Datum“. Allerdings sollte es möglich sein, auch hierzu OpenProposal einzusetzen, indem lediglich ein allgemeiner Kommentar auf das Bildschirmfoto platziert würde, wodurch ein Erfassen des Vorschlagskontextes möglich sei, ohne dass es einen direkten Bezug auf die Benutzeroberfläche geben muss.

Die Bedienbarkeit von OpenProposal wurde als verbesserungswürdig bezeichnet. Das Aktivieren bzw. Deaktivieren von OpenProposal fiel den Anwendern nicht leicht. Außerdem waren die Anwender davon verunsichert, ob sie nach dem Absenden eines Vorschlags auf die Schaltfläche „Neuer Vorschlag“ oder „Vorschlag fortsetzen“ drücken sollen. Weiterhin waren die Anwender mit den Kategorien der Annotationswerkzeuge überfordert. Es konnte beobachtet werden, dass die Anwender zwar

sehr bemüht waren, das passende Werkzeug auszuwählen, aber dabei gleichzeitig verunsichert waren, ob das von ihnen gewählte Werkzeug das richtige war. Nach Meinung der Moderatorin waren die Kategorien der Annotationswerkzeuge aus Entwicklersicht und nicht aus Anwendersicht entwickelt, so dass sie forderte, die Annotationswerkzeuge intuitiv verständlich und einfach bedienbar zu gestalten. Die vorhandenen Schaltflächen wurden prinzipiell als geeignet jedoch als zu zahlreich eingeschätzt. Es wurde angeraten auf detailliierte Annotationswerkzeuge wie z.B. das Hinzufügen eines Pulldownmenus zu verzichten.

In der Nachbereitung der Anwenderbeiträge waren die OpenProposal Beiträge verständlich, konstruktiv und konkret. Für die Moderatorin waren vor allem die Verständlichkeit und die Konstruktivität wichtig. Von den Entscheidern und Entwicklern hatte sie zudem keine Rückfragen erhalten. Einziger Aufwand ergab sich bei der Suche und Auflösung von doppelten Vorschlägen. Dabei habe es sich als sinnvoll erwiesen, dass jeder erstellte Anwenderbeitrag genau eine Annotation bzw. einen Kommentar enthält, da dies die weitere Bearbeitung erleichtert. Es wurde darüber hinaus als interessant eingeschätzt, alle Anwenderbeiträge gleichzeitig anzeigen zu könnten, um eine bessere Übersicht über alle Anwenderbeiträge zu erhalten. Allerdings existierte bis dahin keine Vorstellung, wie dies genau aussehen könnte.

Außerdem sollte davon ausgegangen werden, dass Anwenderbeiträge nicht immer verständlich formuliert werden. Oft lasse sich der Kontext des Beitrags nicht mehr nachvollziehen. So war es häufig für die Moderatorin nicht eindeutig, wann ein Vorschlag erstellt wurde, so dass dieser nicht den gestellten Aufgaben zugeordnet werden konnte. Hierfür fand die Moderatorin die Diskussionen in der gemeinsamen Runde nach den Einzelinterviews und die spätere Analyse der Videos hilfreich, da sie sich nochmals versichern konnte, wie die Anwenderbeiträge gemeint waren. Dasselbe gilt für das Auftreten von Verständnisproblemen. Daher sollte der Anwender bei OpenProposal gebeten werden, auch die Nummer der Testaufgabe beim Erstellen eines Anwenderbeitrages einzugeben und zusätzlich anzugeben, welche Motivation hinter dem jeweiligen Vorschlag steckte.

Als ergänzende Funktion hielt es die Moderatorin für sinnvoll, den Anwendern zum einen ihre eigenen bereits erstellten Beiträge anzuzeigen und ihnen zum anderen auch die von den anderen Anwendern erstellten Anwenderbeiträge zugänglich zu machen. Allerdings sollten nur diejenigen Anwenderbeiträge angezeigt werden, die zur gerade geöffneten Anwendung passen würden. Hier sei es auch hilfreich, wenn Anwender die Vorschläge der anderen bewerten würden, um die Bedeutungsgrad der einzelnen Vorschläge besser einschätzen zu können.

Die Videoanalyse könne man aus ihrer Sicht nicht mit OpenProposal vergleichen, da sie eine andere Form von Anwenderbeiträgen biete. Die Auswertung der Videos erforderte zwar einen hohen Aufwand, aber dafür konnte sie auch die Reaktionen der Anwender während der Tests im Detail analysieren und im Nachhinein auch Parallelen zwischen den einzelnen Tests finden. Dafür hatte die Moderatorin jedes Video mehrmals vollständig angesehen, Auffälligkeiten in einer Tabelle notiert und daraus Anforderungen und Verbesserungsvorschläge erarbeitet. Aus ihrer Sicht konnten hierbei auch andere Arten von Informationen als mit OpenProposal erfasst werden. Dabei war ihr aber auch aufgefallen, dass die Anwender die Kamera als Protokollant im Laufe des Tests vergessen hatten und damit auch nicht mehr an die Formulierung von Anwenderbeiträgen für die Kamera dachten. Daher war es erforderlich, direkt neben den Anwendern zu sitzen und Notizen zu führen. Hierdurch wurde dem Anwender bewusst, dass er einen Gesprächspartner hatte, dem er Feedback mitteilen sollte.

An dieser Stelle wäre es für die Moderatorin interessant gewesen, wenn sie die Anwenderbeiträge aus den Videoanalysen wiederum mit OpenProposal formulieren könnte, um die von den Anwendern mündlich formulierten Anwenderbeiträge nachzustellen. Ihr war aufgefallen, dass es ihr selbst auch nicht immer einfach fiel, die Anforderungen in reiner Textform zu formulieren. Daher war für sie die Videoanalyse eine hilfreiche Unterstützung, die sie im Rahmen von Einzelinterviews immer anwenden würde. OpenProposal stellte hierfür eine ideale Ergänzung dar.

Im laufenden Betrieb hätte sie sich einen aktiveren Einsatz der Anwender mithilfe von OpenProposal erhofft:

> *„Gerade im realen Betrieb würden sich erst herausstellen, welche Aspekte einer Software verbessert werden müssten. In einem Labortest nimmt man noch in Kauf, dass bestimmte Dinge nicht ideal funktionieren. Wenn man diese aber täglich benutzen muss, fordert man energischer nach Besserung. In dieser Testphase wurde allerdings nur wenige Male auf OpenProposal zurückgegriffen. Bei jedem Mal handelte es sich um größere technische Probleme, die zuerst per Mail an den entsprechenden Entscheider kommuniziert wurden und anschließend für die Fehlersuche mit OpenProposal verständlicher dokumentiert wurden."*

Dass OpenProposal nicht noch häufiger genutzt wurde, könnte ihrer Meinung nach daran liegen, dass die Anwender im Betriebsalltag üblicherweise wenig Zeit hatten und dass die WAVES-Softwareanwendungen auch kein primäres System im Betrieb darstellten. In Gespräch mit den Anwendern wurde einstimmig die Be- bzw. Überlastung durch die tägliche Arbeit als Ursache genannt. Diesem hätte man möglicherweise entgegenwirken können, indem man mit den Anwendern bestimmte Zeiträume zum Testen festgelegt und nachdrücklich per Email und Telefon daran erinnert hätte. Alternativen zu OpenProposal kannte sie nicht. Aus ihrer Sicht unterstützte OpenProposal den Prozess der Erstellung von Anwenderbeiträgen im Betriebsalltag bereits sehr gut. Weder Projekt-Wiki noch JIRA hielt sie für geeignet, da diese von den Anwendern aufgrund ihrer Komplexität nicht benutzt würden.

Bei den Entscheidern hinterließ OpenProposal einen positiven Eindruck. Für sie waren die OpenProposal Beiträge verständlich, nachvollziehbar und in fast allen Fällen hilfreich. Die Nachbearbeitung der Anwenderbeiträge mithilfe von JIRA wurde gut umgesetzt und war für die weitere Bearbeitung wichtig. Hilfreich war auch, dass jedes annotierte Bildschirmfoto genau einen Anwenderbeitrag enthielt, da es die weitere Bearbeitung in JIRA vereinfachte. Insgesamt hatten die OpenProposal Anwenderbeiträge im Vergleich zu den textlich formulierten Vorschlägen ein angenehmeres Erscheinungsbild und hoben sich positiv hervor. Die Vorschläge waren dank der grafischen Darstellung ansprechender als die rein textlichen Formulierungen:

> *„Mit Text könnte ein falscher Ton mit rüber kommen. Eine verbale Beschreibung klingt eher fordernd, die OpenProposal Issues waren eher angenehm, da weniger fordernd, und man hat sich gut gefühlt, auch wenn man es nicht gleich erledigt hat."*

Sie schätzen ein, dass ca. 90% der Vorschläge konstruktive Verbesserungsvorschläge und die restlichen 10% reine Fehlermeldungen waren. Dies fanden sie sehr positiv:

> *„Mit OpenProposal waren die Anwender zu mehr Kreativität motiviert. Dadurch erhielten wir als Hinweise nicht nur Fehler oder ‚das fehlt‘ sondern auch ‚das könnte besser sein wenn es eine andere Form‘ annimmt.“*

Außerdem eigneten sich die OpenProposal Anwenderbeiträge zur Diskussion mit mehreren Leuten, da sie sofort als Bild erfassbar waren und der Kontext des Beitrags durch das Bildschirmfoto erkennbar war. Für sie hatten die OpenProposal Anwenderbeiträge auch mehr Bedeutung, da sie sich von rein textlichen Formulierungen abhoben und „einbrannten“:

> *„Gut war es über das Visuelle alles schnell einzuschätzen und einzuordnen, bei Text dauert es etwas länger, bis man es im System verorten kann“*

Insgesamt fanden alle Entscheider den Einsatz von OpenProposal sinnvoll. Der Aufwand war trotz des Anstieges der Anzahl der Anwenderbeiträge niedrig. Andere mediale Kommunikationsmittel wie Video- oder Audioaufnahme waren ihnen zu aufwändig. Annotierte Bildschirmfotos wurden daher als ein gutes Kommunikationssystem eingeschätzt. Bevor OpenProposal eingesetzt wurde war die Kommunikation mit der Moderatorin aufwändiger und es waren häufiger Rückfragen zu den Anwenderbeiträgen der Anwender notwendig. Die Möglichkeit, JIRA ohne OpenProposal zu benutzen, wurde von den Anwendern im gesamten Projektverlauf selten genutzt. Wenn überhaupt, wurden nur Fehlermeldungen eingetragen. Die Anwenderbeiträge waren zuvor auch deutlich allgemeiner und die detaillierten Ausprägungen mussten von der Moderatorin erst nachträglich von den Anwendern erfragt werden. Die Rückmeldungen über die Moderatorin nach den Anwendertests kamen in den früheren Phasen von WAVES auch deutlich später und sehr unregelmäßig. Mit OpenProposal traf zwar plötzlich eine Vielzahl an Anwenderbeiträge nach einem Test ein, die aber in kurzer Zeit durchgearbeitet und an die Entwickler kommuniziert werden konnten.

Einer der Entscheider meinte darüber hinaus, dass OpenProposal einen ersten Schritt darstellte und zu weiteren Aktivitäten motivierte. Allerdings befürchtete er auch, dass über das Ziel hinausgeschossen werden würde. Dadurch bestünde die Gefahr, dass die Software zu stark den Vorstellungen der Anwender und zu wenig der Vision der Entscheider bzw. Entwickler entsprechen würde:

> *„Es ist klasse, es macht Spaß, es regt zum Denken an, was man noch mehr machen könnte. Aber wenn man da mehr machen würde, könnte es sein, dass es zu viel wird.* Dem Anwender wird vielleicht versprochen, dass er das immer bekommt, was er sich wünscht. *Der User darf gerne was äußern, aber wichtiger ist die Gesamtvision des Entwicklers.*“

Als mögliche Weiterentwicklung würde sich eine stärkere soziale bzw. kollaborative Komponente nach Meinung eines Entscheiders eignen, bei der die Anwender ihre Ideen und Vorschläge gegenseitig betrachten und diskutieren könnten. Dabei sollte den Anwendern bei der Benutzung der Software auf der grafischen Oberfläche proaktiv gezeigt werden, dass zu bestimmten Bereichen bereits Vorschläge anderer Anwender existierten.

Allerdings wurde die Nützlichkeit der Anwenderbeiträge von den Entscheidern insgesamt in Frage gestellt. Einer der Entscheider merkte an, dass der Entwicklung ihrer Software klare Vorstellungen der Entscheider zugrunde liegen. Das übliche allgemeine Feedback der Anwender würde häufig den eigenen Vorstellungen entsprechen und damit nur wenig Neues einbringen:

> *„Wir wussten oft was wir sowieso machen wollten, daher war das allgemeine Feedback der Anwender nur teilweise hilfreich."*

Bei Vorschlägen, die den eigenen Vorstellungen widersprachen, hatten sich die Entscheider meistens vorher schon Gedanken gemacht und sich bewusst dagegen entschieden. Daher wurde ein detailliertes Feedback der Anwender als hilfreicher eingeschätzt, da die konkrete Gestaltung der Umsetzung der Anwenderwünsche oft nicht klar war. Zu jedem Anwenderbeitrag wäre es für die Entscheider außerdem interessant zu erfahren, wie viele andere Anwender diesem zustimmen würden, um die Einteilung der Prioritäten verbessern zu können. Insgesamt spielte aus Sicht der Entscheider vor allem die Diskussion und Konsensfindung mit den Anwendern eine große Rolle, um den Anwender an der Software-Entwicklung partizipieren zu lassen und ihn über den Hintergrund der Entscheidungen zu informieren:

> *„Man kann nicht alle glücklich machen, daher Diskussion und Moderation wichtig um einen Konsens zu gewinnen."*

Von den OpenProposal Anwenderbeiträgen wurden zwar viele Vorschläge für sinnvoll gehalten, aber es konnten nicht alle umgesetzt werden. Dies lag daran, dass die Zeit und die Ressourcen nicht mehr ausgereicht hatten und die Anwenderbeiträge somit für ein späteres Release eingeplant wurden. Das Projekt WAVES wäre zwar beendet, aber die die Arbeiten würden in anderen Forschungsarbeiten fortgesetzt und es sollen nachträglich noch Aktualisierungen an die Anwender bereitgestellt werden. Von den Anwendern gab es aber dazu auch keine Rückfragen mehr. In dem Abschlusstreffen von WAVES schienen die Anwender mit dem Stand der Entwicklungen zufrieden und mit der Planung der Entscheider ohne Einwände einverstanden zu sein.

Es wurden von den Entscheidern jedoch auch kritische Anmerkungen abgegeben. Die OpenProposal Beiträge adressierten nie etwas Größeres. Außerdem wurde es als ausreichend bezeichnet, wenn der Anwender an der richtigen Stelle markiert, was er sich wünschte. Eine Kategorisierung der Anwenderbeiträge wurde als nicht unbedingt notwendig erachtet. Bei Fehlermeldungen sollten zukünftig auch die Fehlerberichte („log Dateien") beigefügt werden.

Ein weiteres Manko stellt der Titel dar. Diese waren häufig ungünstig gewählt und hatten kaum Aussagekraft. Der Titel wurde aber oft so belassen, um dem Anwender später eine Wiedererkennung und eine Nachverfolgung der Bearbeitung zu ermöglichen. Außerdem wurde hervorgehoben, dass die OpenProposal Anwenderbeiträge keine Anforderungen oder Spezifikationen darstellten. Daher wurde OpenProposal zwar als hilfreich eingeschätzt, um Anwender besser einzubeziehen, allerdings schien es nicht dazu geeignet, eine klassische Anforderungsanalyse zu ersetzen.

Als wünschenswert wurde zudem eine bessere Übersicht über die Anwenderbeiträge angemerkt. Dass jedes annotierte Bildschirmfoto genau einen Beitrag enthielte, wurde zwar als sehr sinnvoll erachtet, aber möglicherweise wäre es auch hilfreich, wenn alle Vorschläge zu einer Software-Komponente nebeneinander gelegt werden könnten, um Widersprüche zwischen den Anwenderbeiträgen finden zu können. Spannend wäre es dann, wenn man bei Auftreten von unterschiedlichen sich widersprechenden Vorschlägen eine Rückfrage an alle Anwender stellen und die beste Variante bewerten lassen würde.

Aus Sicht der Entwickler waren die OpenProposal Anwenderbeiträge verständlich und stellten eine Besserung der vorherigen Situation dar. Allerdings war auch die vorherige Situation bereits sehr gut und die Kommunikation zwischen Entscheider und Entwickler verlief über JIRA bestens.

> *„Es gab es wöchentliche Treffen, aber die meiste Kommunikation lief über JIRA direkt und weniger über direkte Gespräche. Die Issues waren aber auch immer klar verständlich und daher war klar, was zu tun war. Die Kommunikation über JIRA lief reibungslos. Die Aufgaben waren gut formuliert. Hilfreich ist immer, wenn Reproduktionsschritte aufgezeigt wurden.“*

Die Entscheider hätten die Anwenderbeiträge zuerst erhalten und zu den Beiträgen Anweisungen für die Umsetzung hinzugefügt. Somit wurden die Anwenderbeiträge durch die Entscheider für den Entwickler übersetzt. Die annotierten Bildschirmfotos waren grafisch ansprechend und hätten zudem visuell den Kontext des Vorschlags geliefert. Allerdings waren die Bildschirmfotos nicht immer hilfreich. Vor allem bei Fehlermeldungen war es trotz Bildschirmfoto und Annotation nicht einfach, den Fehler zu reproduzieren:

> *„Bild sehr hilfreich: TOP! Wenn man mit dem Bild etwas anfangen kann. War aber nicht immer so: Bild war da, Annotation war auch vorhanden, Heike[37] hat sozusagen übersetzt und im JIRA notiert, was zu tun ist. Es wäre wohl gut gewesen, die Anwender länger zu schulen, damit sie wissen, wie man Vorschläge formuliert, vor allem auch die Reproduktionsschritte bei Fehlermeldungen angibt.“*

Zur Reproduktion der Fehlermeldungen wären Protokolldaten interessant gewesen. Die Vorschläge, die sich auf die Bedienoberfläche bezogen, waren mithilfe der Bildschirmfotos sehr gut dokumentiert. Es gab aber auch Probleme, bei denen Protokolldaten nicht helfen konnten. Erst ein Besuch der Entwickler vor Ort führte in wenigen Minuten zur Lösung.

Bezüglich des Aufwands meinten die Entwickler, dass die Bildschirmfotos keinen zusätzlichen Aufwand darstellten und als angenehm wahrgenommen wurden:

> *„Bilder sagen mehr als 1000 Worte.“*

Der Text in den Annotationen wurde allerdings als nicht hilfreich gewertet. Zum einen waren die Anmerkungen der Anwender zu oberflächlich und es war häufig nicht nachvollziehbar, was die Anwender genau meinten. Zum anderen enthielten die Anweisungen vom Entscheider bereits ausführlichere und klarere Informationen. Ein Entwickler merkte dazu an:

> *„Dem Anwender ist nicht klar, dass ein Entwickler nicht weiß, was der Anwender genau meint.“*

Der mit OpenProposal gewählte Ansatz war aus ihrer Sicht angemessen und ansprechend. Ihrer Meinung nach wurden den Anwendern jedoch durch die Formulierung von Freitext bereits zu viele Freiheiten zugestanden. Sie räumten gleichzeitig auch ein, dass zu enge Strukturen die Akzeptanz der Anwender mindern könnte. Für sie wären die drei Kategorien „Hinzufügen“, „Verschieben“ und

[37] Name von einem der Entscheider aus Gründen der Anonymisierung geändert

„Verbessern“ für die Annotationswerkzeuge ausreichend. Zudem würden sie sich mehr Kontextdaten wie z.B. Anmeldenamen, Datum, Version des Internetbrowsers, Konfiguration des Betriebssystems, Version der Java Runtime, Verzeichnisstruktur der Programme im Betriebssystem, etc. wünschen:

> *„Mehr Daten ist immer gut."*

Andere Kommunikationsmedien wie Video oder Audio hielten sie nicht für hilfreich. Dies wäre für sie ein zusätzlicher Aufwand, von dem sie sich keinen Vorteil versprechen würden:

> *„Ich glaube nicht, dass es mehr bringen würde, da er [der Anwender] wohl auch nicht die richtigen Hinweise geben kann, wenn er das nicht schon über ein Bildschirmfoto zeigt.“*

9.5.4 Diskussion der Ergebnisse der Fallstudie „WAVES“

OpenProposal hinterließ bei allen Beteiligten einen positiven Gesamteindruck. Die Ergebnisse aus der ersten Fallstudie ließen sich in fast allen Aspekten replizieren. Von den Anwendern wurde OpenProposal als einfache und ansprechende Methode wahrgenommen. Die Moderatorin sah den großen Vorteil, dass die Anwender zu konkreteren und konstruktiveren Anwenderbeiträgen motiviert werden konnten. Die Entscheider waren vor allem über die zeitnahe Zusendung von konkretem und konstruktivem Feedback erfreut und hielten die Kommunikation mit den Anwendern mit OpenProposal für angenehm und effizient. Die Entwickler fanden die OpenProposal Beiträge im Gesamten verständlich und hilfreich.

OpenProposal wurde bei dem ersten Unternehmen wider Erwarten kein einziges Mal verwendet. Dies ist nach Ansicht der Moderatorin auf die ungewohnte Situation der Anwender zurückzuführen, zuerst mit der „Thinking aloud“ Methode die Idee mündlich auszusprechen und anschließend nochmals schriftlich mit OpenProposal zu formulieren. So merkte sie an, dass sie die Benutzung von OpenProposal beim ersten Unternehmen lediglich mit niedriger Priorität lanciert hatte. Erst eine stärkere Motivation durch die Moderatorin in den folgenden Einzelinterviews, die Vorschläge nicht nur mündlich sondern auch schriftlich zu formulieren, führte zu einem aktiveren Einsatz von OpenProposal. Die Einflussnahme der Moderatorin wurde von keinem der Anwender als störend empfunden. Es stellt sich die Frage, ob auch während des Workshops in der ersten Fallstudie eine häufigere Motivation zur Benutzung von OpenProposal zu einer höheren Anzahl an Vorschlägen geführt hätte.

Die Anwender, die OpenProposal verwendeten, verinnerlichten das Bedienkonzept trotz kleiner Mängel in der Verständlichkeit innerhalb kurzer Zeit. Die grafische Benutzeroberfläche und das Konzept der Annotation von Bildschirmfotos wurden als sehr gut bewertet. Dem weiteren Einsatz von OpenProposal stimmten alle Anwender zu. Als mögliche Verbesserungen wurde vorgeschlagen, OpenProposal um neue Funktionen zu erweitern, u.a. Tastaturkürzel zuzulassen, vorgegebene Grafiken bzw. Icons zum Hinzufügen anzubieten sowie bestehende Bereiche des Bildschirmfotos kopieren und einfügen zu können. Wie bereits in Fallstudie 1 wurde die Frage aufgeworfen, wie Abläufe über mehrere Software-Bereiche in einem Vorschlag annotiert werden können. Die Moderatorin äußerte zudem den Wunsch nach einer besseren Übersicht bei der Verwaltung der annotierten Bildschirmfotos. Außerdem schlug sie vor, den Anwendern ihre eigenen Vorschläge anzuzeigen.

Die Analyse der abgegebenen Vorschläge ergab, dass die OpenProposal Vorschläge zwar durchgängig kategorisiert wurden, aber die gewählten Kategorien nur in 49 Fällen sinnvoll gewählt waren. In 26 Fällen wäre hingegen anstelle der Kategorie „Kommentar“ eine andere Kategorie sinnvoller gewesen.

Dabei war auffällig, dass der allgemeine „Kommentar“ nur in vier von 30 Fällen berechtigt war und als Zeitersparnis bei der Suche nach der passenden Kategorie vom Anwender favorisiert wurde. Der Kategorisierung wurde allerdings weder von Entscheidern noch von Entwicklern eine große Bedeutung beigemessen. Ihnen waren vor allem ein aussagekräftiger Titel und eine schriftliche Kurzbeschreibung zum annotierten Bildschirmfoto wichtig.

Die Analyse der Anwenderbeiträge ergab weiter, dass die OpenProposal Vorschläge durchschnittlich 14,58 Wörtern enthielten. In den Videoaufzeichnungen hingegen wurde jeder Vorschlag mit zahlreichen Sätzen ausführlich kommentiert, damit war entsprechend die Anzahl der verwendeten Wörter deutlich höher. Dies bestätigt die Annahme, dass die Videoaufnahmen reichhaltiger sind und die OpenProposal Anwenderbeiträge ergebnisorientiert die Zusammenfassung der Einzelinterviews enthalten. Somit ist der Umfang der Videoanalyse deutlich höher als bei OpenProposal.

Die Konstruktivität der Anwenderbeiträge ist bei den OpenProposal Beiträgen höher. Dies kann damit begründet werden, dass OpenProposal den Anwender zu ausführlicheren und konkreteren Beschreibungen motiviert. Somit kann den OpenProposal Anwenderbeiträgen eine hohe Konstruktivität zugeschrieben werden. Bei den OpenProposal Anwenderbeiträgen lag der Anteil der Problemberichte bei 12,50% (10 von 83), bei den Videoanalysen bei 71,43% (10 von 13). Folglich wurden mit OpenProposal deutlich mehr konstruktive Anwenderbeiträge abgegeben. Dabei erstellten 13 der 16 Anwender einen oder mehr Anwenderbeiträge mit OpenProposal. Die Verständlichkeit der OpenProposal Beiträge wurde als hoch bewertet. Eine Bewertung der Verständlichkeit der Anwenderbeiträge aus den Videoaufnahmen war nicht möglich, da die Moderatorin bei der Nacharbeitung eigene Formulierungen aus den Anwenderbeiträgen der Anwender generierte und Entscheider und Entwickler keinen Zugriff auf die ursprünglichen Beiträge hatten.

Im laufenden Betrieb gaben die Anwender abgesehen von OpenProposal, mit dem acht Anwenderbeiträge erstellt wurden, mit keiner der anderen Methoden Anwenderbeiträge ab. Von den Anwendern war keine andere Erklärung zu erhalten, als dass sie zu sehr mit ihrem Tagesgeschäft beschäftigt waren und keine Zeit für die Formulierung von Anwenderbeiträgen hatten. OpenProposal selbst wurde kein Fehler zugeschrieben. Verbesserungsmöglichkeiten konnten weder von den Anwendern noch von der Moderatorin identifiziert werden.

In Bezug auf die abhängigen Variablen AV1 - 10 konnte in der Fallstudie eine aussagekräftige Datenbasis geschaffen werden, mit der Antworten auf die Hypothesen H1 - H5 gegeben werden können. Tabelle 9.7 setzt die Ergebnisse der Fallstudie in Bezug zu den abhängigen Variablen AV1 - AV10 und der unabhängigen Variable UV1 mit UV1-1 OpenProposal und UV1-4 Videoanalyse. Keine Erwähnung finden UV1-2 JIRA, UV1-3 Projekt-Wiki und UV1-5 Feedbackformular, da die Anwender mit diesen keine Beiträge erstellten.

Für die Untersuchung der Hypothesen konnten dabei folgende Feststellungen gemacht werden:

1. Mit einer vielfach höheren Anzahl von Anwenderbeiträgen bei OpenProposal kann Hypothese H1 unterstützt werden.

2. Hypothese H2 kann unterstützt werden, da die mit OpenProposal erstellten Vorschläge eine sehr hohe Konstruktivität zugeordnet wurden.

3. Mit OpenProposal wurden über eine Laufzeit von zwei Monaten acht Anwenderbeiträge im Betriebsalltag erstellt. Es wurden nicht noch mehr Beiträge abgegeben, da die getesteten Forschungsprototypen keine zentrale Bedeutung für die Anwender darstellten. Hypothese H3 kann soweit bestätigt werden, als dass OpenProposal im Unterschied zu den anderen Methoden eine Hilfe für die Kommunikation zwischen Anwender und Entwickler im Falle von kritischen Anforderungen darstellte. Zudem war die Zufriedenheit der Anwender mit OpenProposal hoch und der Aufwand und die Komplexität zur Erstellung von Anwenderbeiträgen niedrig.

4. Im Vergleich zur Videoanalyse war mit OpenProposal nur ein geringer Aufwand zur Vor- und Nachbereitung notwendig. Dies unterstützt Hypothese H4.

5. Die hohe Verständlichkeit der OpenProposal Beiträge unterstützt Hypothese H5.

In der zweiten Fallstudie „WAVES“ konnten somit die Ergebnisse aus der ersten Fallstudie „TRUMPF“ repliziert werden. Hypothese H3 kann zwar unterstützt werden, aber dabei zeigte sich, dass die Methode für sich alleine nicht ausreicht sondern auch von der Wichtigkeit der Software für die Anwender abhängt. Positiv konnte festgehalten werden, dass die Unternehmen CAS, PTV und Disy an einem weiteren Einsatz von OpenProposal in anderen Projekten stark interessiert waren.

Die erfassten Verbesserungsideen zu OpenProposal wurden im Entwicklungsteam von OpenProposal diskutiert und bis auf den Vorschlag zum Tastaturkürzel verworfen, da diese die Komplexität von OpenProposal erhöht hätten und nur für spezielle Einzelfälle praktikabel gewesen wären. Die Kategorien der Annotationswerkzeuge erfuhren eine Überarbeitung, da diese vom Anwender nicht korrekt angewendet wurden. Außerdem wurde nach den Vorschlägen der Entscheider eine automatisierte Generierung von Titel und Kurzbeschreibung implementiert.

Die Fallstudie zeigte darüber hinaus auch Limitierungen für den Einsatz von OpenProposal. Anwenderbeiträge, die mehrere Anwendungsfenster gleichzeitig betreffen, sind mit OpenProposal erschwert zu formulieren. Außerdem wird stets eine grafische Repräsentation in Form eines Prototypen der zu verbessernden Software benötigt. Insgesamt ist OpenProposal somit nur für diejenigen Einsatzszenarien sinnvoll, bei denen Anwender die grafische Bedienoberfläche einer Software an einem Computer testen können und bei denen in erster Linie konkrete und konstruktive Anwenderbeiträge gewünscht sind.

Ferner weist die Kombination von OpenProposal und Videoanalyse mehrere Vorteile auf, obwohl OpenProposal ursprünglich für eine asynchrone Anwenderbeteiligung geplant war. Während mit OpenProposal direkt nach der Anwenderbefragung Verbesserungsvorschläge an Entscheider und Entwickler weitergegeben werden können, kann mit der Videoanalyse ausführlich auf Details der Anwenderkommentare geachtet werden. So steht die Moderatorin bei der Analyse der Videoaufnahmen nicht unter Zeitdruck, da sich die Entwickler dank OpenProposal bereits zeitnah mit den wichtigsten Ergebnissen der Befragungen beschäftigen können und nicht auf die Nachbereitung der Videoanalyse warten müssen. Außerdem ergeben sich aus den beiden Werkzeugen unterschiedliche Formen von Anwenderbeiträgen. Mit der Videoanalyse erläutern die Anwender ihre Vorschläge sehr ausführlich und im Allgemeinen eher abstrakt. OpenProposal führt hingegen zu konkreten aber eher knapp formulierten Vorschlägen.

AV	UV1-1 OpenProposal	UV1-4 Videoanalyse
AV1: Anzahl der erstellten Anwenderbeiträge	Hoch (Interviews) 75 Mittel (Betriebsalltag) 8	Niedrig 13
AV2: Umfang der erstellten Anwenderbeiträge	Mittel (Interviews) 14,57 Wörter pro Vorschlag und 75 Bildschirmfotos Niedrig (Betriebsalltag) 3,00 Wörter pro Beitrag und 8 Bildschirmfotos	Hoch 1 Stunde Aufnahme pro Interview
AV3: Konstruktivität der erstellten Anwenderbeiträge	Hoch 73 konstruktiv	Niedrig 3 konstruktiv
AV4: Aufwand des Anwenders zur Erstellung von Anwenderbeiträgen	Niedrig	Niedrig
AV5: Komplexität der Erstellung von Anwenderbeiträgen	Niedrig	Niedrig
AV6: Zufriedenheit der Anwender mit der Methode zur Anwenderbeteiligung	Hoch 1,88	Mittel 2,43
AV7: Aufwand des Moderators für die Vorbereitung zur Erfassung von Anwenderbeiträgen	Niedrig	Niedrig
AV8: Aufwand des Moderators für die Nachbereitung der Anwenderbeiträge	Niedrig 8,00% Änderungen	Hoch 0,00% Änderungen
AV9: Aufwand des Moderators für die Interaktion mit den Anwendern	Hoch (Interviews) Niedrig (Betriebsalltag)	Hoch
AV10: Verständlichkeit der erstellten Anwenderbeiträge	Hoch 1,67: Trifft größtenteils zu	Hoch (keine Benotung)

Tabelle 9.10: Zusammenfassung der Ergebnisse aus der Fallstudie „WAVES“

Die Werkzeuge Feedbackformular, Projekt-Wiki und JIRA fanden keine Anwendung. Bei dem Projekt-Wiki und bei JIRA wurde angemerkt, dass die Einstiegsbarriere als zu hoch empfunden wurde, da die Bedienung der beiden Methoden eine gute Orientierung erfordert und die textliche Beschreibung der Vorschläge als aufwändig eingeschätzt wurde. Zum Feedbackformular konnte keine Erklärung erfasst werden.

9.6 Fallvergleichende Analyse und Interpretation

Die fallvergleichende Betrachtung beider Fallstudien bestätigt die Hypothesen H1, H2, H3, H4 und H5. Die Erfahrungsberichte und Analysen der Verbesserungsvorschläge sind sich in beiden Fallstudien sehr ähnlich und legen die Vermutung nahe, dass sich in weiteren Studien ähnliche Erfahrungen ergeben würden. Der Einsatz von OpenProposal wurde von allen Teilnehmern begrüßt und als sinnvolle Erweiterung zu existierenden Methoden geschätzt.

Bezugnehmend auf die eingangs aufgestellten Forschungsfragen F1, F2 und F3 können folgende Antworten gegeben werden:

F1: Wird OpenProposal von Anwendern und Moderatoren zur Erstellung von Anwenderbeiträge in Software-Projekten akzeptiert und wenn ja weshalb?

Die Akzeptanz von OpenProposal konnte bei Anwendern und Moderatoren nachgewiesen werden. Die Gründe liegen hauptsächlich in der einfachen und schnellen Bedienung und in der Integration in die Projektumgebung. Die Möglichkeit zur Annotation von Bildschirmfotos erlaubt es dem Anwender seinen Anwenderbeitrag mit wenigen Handgriffen zu formulieren und abzugeben. Für den Moderator erweist sich die elektronische Datenhaltung in einem Issue-Tracker als vorteilhaft bei der Nachbereitung der Anwenderbeiträge.

F2: Wird OpenProposal von Moderatoren, Entscheidern und Entwicklern in Software-Projekten akzeptiert und wenn ja weshalb?

OpenProposal fand bei Moderatoren, Entscheidern und Entwickler hohe Akzeptanz bei der Bearbeitung der Anwenderbeiträge. Entscheidend für sie war, dass die mit OpenProposal erstellten Beiträge verständlich und konkret sind. Außerdem ist es von Vorteil, dass sie nach ihrer Erfassung zeitnah an Entscheider und Entwickler kommuniziert werden können.

F3: Welche Vor- und Nachteile hat OpenProposal gegenüber anderen Methoden?

In den Fallstudien konnten unterschiedliche Methoden mit OpenProposal verglichen werden. Im direkten Vergleich zu OpenProposal standen die Methoden JIRA, Videoanalyse, Feedbackformular und Projekt-Wiki. Außerdem wurden Fragebögen und automatisierte Problemberichte eingesetzt, die einen anderen Zweck verfolgten, aber auch zur Kategorie der asynchronen Methoden zählen. Im Vergleich zu diesen Methoden erweist sich OpenProposal in vielfacher Hinsicht als vorteilhaft.

Mit OpenProposal wurden mehr Anwenderbeiträge erstellt, zudem wurde die Verständlichkeit der mit OpenProposal erstellten Anwenderbeiträge besser bewertet, und auch die Zufriedenheit der Anwender war mit OpenProposal höher. Als Vorteil wurde die Einfachheit des grafischen Annotierens hervorgehoben, mit dem die Ideen und Vorstellungen der Anwender verständlich und konkret formuliert werden können. Dem Anwender wird die Beschreibung seines Beitrages erleichtert, da er seinen Wunsch mithilfe der Annotationswerkzeuge grafisch äußern kann. Außerdem wird er dazu motiviert, sich auf das Bildschirmfoto zu beziehen und seinen Beitrag konkret darzustellen.

In der zweiten Fallstudie konnte ein Vergleich mit der Videoanalyse durchgeführt werden. Für die Anwender war es einfacher, ihre Beiträge mündlich laut auszusprechen und sich von der Videokamera aufnehmen zu lassen. Mit OpenProposal waren sie aufgrund der schriftlichen Formulierung gezwungen ihren Beiträge knapp und konkret zu beschreiben. Während mit OpenProposal direkt nach der Befragung die Ergebnisse vorliegen, erfordert die Videoanalyse hingegen einen hohen Aufwand für die Auswertung der Videos und der Ableitung der konkreten Anwenderbeiträge.

Ein weiterer Vorteil ist, dass die Anwender mit OpenProposal zur Abgabe von konstruktiven Anwenderbeiträgen motiviert werden. So konnte in den beiden Fallstudien beobachtet werden, dass Anwender mit OpenProposal überwiegend konstruktive Verbesserungsvorschläge und anteilsmäßig weniger Fehlermeldungen formulierten.

In beiden Fallstudien wurden Fragebögen verwendet. Allerdings wurden diese nicht zur Erfassung von Anwenderbeiträgen, sondern zur Erfassung der Zufriedenheit mit dem getesteten Prototypen und dem Ablauf des Testes eingesetzt. Von keinem der Moderatoren wurde der Fragebogen als Methode zur Erfassung von Verbesserungsvorschlägen in Erwägung gezogen.

In der ersten Fallstudie wurde die Möglichkeit zur Nutzung eines Bildschirmfotoprogramms gewährt. Dies war eine in JIRA integrierte Funktion. Allerdings wurde diese Funktion nur von einer Person verwendet. Die Bedienung und die mangelnde Ausdrucksfähigkeit konnten als hauptsächliche Ursachen hierfür identifiziert werden.

In der zweiten Fallstudie wurde die Möglichkeit zum automatisierten Problembericht genutzt. Hierfür wurde bei dem getesteten Prototyp WavesIS die Nutzerprotokollierung (z.B. Seitenaufrufe, Übertragung von Daten) eingerichtet, um Fehlerberichte besser analysieren zu können. Allerdings konnte dies nicht gewinnbringend für die Ableitung von Anwenderbeiträgen verwendet werden.

Ferner wurde OpenProposal zur Erstellung von Anwenderbeiträgen im Rahmen der täglichen Arbeit eingesetzt. Dies zeigt, dass eine asynchrone Anwenderbeteiligung mit OpenProposal möglich ist. Allerdings zeigt die Fallstudie auch, dass auch eine für ideal bewertete Methode nur eine geringe Anzahl an Anwenderbeiträgen liefert, wenn die zu testende Software keine zentrale Bedeutung für die Anwender besitzt.

Neben den genannten Vorteilen stößt der OpenProposal-Ansatz allerdings auch auf zahlreiche Limitierungen: Alle Teilnehmer waren sich einig darin, dass das Annotieren von Bildschirmfotos in manchen Fällen nicht die am besten geeignete Methode sei. Als Ergänzung zu anderen Methoden wie z.B. Interviews mit Videoanalysen und Usability Workshops wurde OpenProposal befürwortet.

Darüber hinaus wurde ersichtlich, dass der Einsatz von OpenProposal von der Motivation des Moderators abhängt. Eine regelmäßige und häufige Erinnerung des Anwenders an die Benutzung von OpenProposal ist notwendig. Die Anwender störten die Erinnerungen bzw. Hinweise nicht.

9.7 Fazit zu Kapitel 9

Ziel von Kapitel 9 war die Evaluation der OpenProposal Methode. Hierfür erwies sich die Methode der vergleichenden Fallstudien als am besten geeignet. Die Fallstudien wurden geplant, deren Verlauf beschrieben und die Ergebnisse zusammengefasst. In der ersten Fallstudie wurde OpenProposal im Rahmen der Software-Entwicklung bei dem Unternehmen TRUMPF in einem Usability Workshop eingesetzt. Die Fallstudie 2 befasste sich mit dem Einsatz von OpenProposal in der Software-Entwicklung im Rahmen des Forschungsprojekts WAVES. Den Fallstudien wurden die Forschungsfragen F1 - F3 zugrunde gelegt:

F1: **Wird OpenProposal von Anwendern und Moderatoren zur Erfassung von Verbesserungsvorschlägen in Software-Projekten akzeptiert und wenn ja weshalb?**

F2: **Wird OpenProposal von Moderatoren, Entscheidern und Entwicklern zur Bearbeitung von Verbesserungsvorschlägen in Software-Projekten akzeptiert und wenn ja weshalb?**

F3: **Welche Vor- und Nachteile hat OpenProposal gegenüber anderen bereits existierenden Werkzeugen?**

Zu Beantwortung der Forschungsfragen wurden die Hypothesen H1 - H5 definiert:

H1: **Mit OpenProposal werden im Vergleich zu anderen Methoden mehr Anwenderbeiträge erfasst.**

H2: **Der Einsatz von OpenProposal führt zu konstruktiven Anwenderbeiträgen.**

H3: **OpenProposal motiviert Anwender dazu, Beiträge im Betriebsalltag zu erstellen.**

H4: **OpenProposal reduziert den Aufwand der Anwenderbeteiligung.**

H5: **Die Verständlichkeit der mit OpenProposal erstellten Anwenderbeiträge ist hoch.**

Zur Untersuchung der Hypothesen wurde die unabhängige Variable UV1 definiert, mit der in den Fallstudien die Werkzeugwahl beschrieben wird:

UV1: **Wahl der Methode zur Anwenderbeteiligung**

Entsprechend der Wahl der unabhängigen Variable UV1 wurden anhand der abhängigen Variablen AV1 - AV10 die Effekte der Werkzeugwahl untersucht:

AV1: Anzahl der erstellten Anwenderbeiträge
AV2: Umfang der erstellten Anwenderbeiträge
AV3: Konstruktivität der erstellten Anwenderbeiträge
AV4: Aufwand des Anwenders zur Erstellung von Anwenderbeiträgen
AV5: Komplexität der Erstellung von Anwenderbeiträgen
AV6: Zufriedenheit der Anwender mit der Methode zur Anwenderbeteiligung
AV7: Aufwand des Moderators für die Vorbereitung zur Erfassung von Anwenderbeiträgen
AV8: Aufwand des Moderators für die Nachbereitung der erstellten Anwenderbeiträge
AV9: Aufwand des Moderators für die Interaktion mit den Anwendern
AV10: Verständlichkeit der erstellten Anwenderbeiträge

Die Planung und Durchführung beider Fallstudien orientierte sich an diesen Variablen. In der ersten Fallstudie wurde der Einsatz von OpenProposal im Rahmen eines zwei-tägigen Usability Workshops mit elf Teilnehmern untersucht und einem Vergleich mit dem Issue-Tracker JIRA unterzogen. In der zweiten Fallstudie wurden neben OpenProposal und JIRA die Werkzeuge Videoanalyse, Feedbackformular und Projekt-Wiki als Vergleichsmethoden hinzugezogen.

Die Auswertung der Fallstudien basierte auf Fragebögen, Interviews, Beobachtungen und der Analyse der erstellten Anwenderbeiträge. Folgende Feststellungen konnten in den Fallstudien gemacht werden:

- Die Hypothesen H1, H2, H3, H4 und H5 konnten bestätigt werden. Im Vergleich zu den anderen Methoden wurden mit OpenProposal mehr Anwenderbeiträge abgegeben, die Anwenderbeiträge waren sehr konkret, konstruktiv und gut verständlich. Mit OpenProposal wurden auch im Betriebsalltag Anwenderbeiträge abgegeben, allerdings blieb deren Anzahl hinter den Erwartungen zurück. Der Aufwand zur Anwenderbeteiligung mit OpenProposal wurde als gering eingestuft.

- Von Anwendern und Moderatoren wurde OpenProposal als sinnvoll und hilfreich sowohl für eine synchrone als auch für eine asynchrone Anwenderbeteiligung bewertet.

- Moderatoren, Entscheider und Entwickler befanden die mit OpenProposal erstellten Anwenderbeiträge als verständlich und hilfreich. Außerdem fanden sie es von Vorteil, dass die Anwenderbeiträge direkt nach der Anwenderbefragung bereitstanden. Lediglich die Beschreibung der Titel der Anwenderbeiträge wurde als verbesserungswürdig eingeschätzt, wobei hier eine automatisierte Generierung der Titel Abhilfe schaffen könnte.

- Das Unternehmen TRUMPF setzte OpenProposal in weiteren Projekten ein. Die an der Fallstudie WAVES beteiligten Unternehmen waren ebenfalls an einer weiteren Verwendung interessiert.

- Für den Einsatz von OpenProposal ist das Vorhandensein eines Prototypen notwendig. Außerdem eignet sich OpenProposal nicht dazu, Anwenderbeiträge zu allgemeinen Anforderungen bzw. nicht-funktionalen Anforderungen zu erheben.

In diesem Kapitel konnten dabei folgende eigene Beiträge herausgearbeitet werden:

- Im Rahmen der Evaluation konnte gezeigt werden, dass vergleichende Fallstudien für die Evaluation von neuen Methoden zur Anwenderbeteiligung geeignet sein können. Dabei wurde demonstriert, wie Fallstudien geplant und Ergebnisse fallstudienübergreifend ausgewertet werden können.

- Für die Fallstudien wurden abhängige Variablen definiert, die als Messgrößen zur Bewertung von Methoden zur Anwenderbeteiligung herangezogen werden können. Diese Variablen stellen ausgehend von der Ableitung der Hypothesen und der Forschungsfragen der Fallstudien die Operationialisierung der Bewertungskriterien aus Kapitel 2.9 dar und können für andere Fallstudien angewendet werden. Diese Variablen können auch als Vergleichskriterien für weitere Untersuchungen und die Entwicklung anderer Methoden zur Anwenderbeteiligung herangezogen werden.

- Die Auswertung der Fallstudien diente der Überprüfung der Hypothesen und der Beantwortung der eingangs aufgestellten Forschungsfragen F1, F2 und F3. Sie führte schließlich zu einer Bewertung von OpenProposal und zur Beantwortung der vierten Forschungsfrage dieser Arbeit nach ihren Möglichkeiten zur Verbesserung der Effizienz der Entwicklung, der Qualität der Anwenderbeiträge und der Belastung der Anwender. Die positive Bestätigung der Hypothesen zeigt auf, wie mit OpenProposal die Nachteile der existierenden Methoden zur asynchronen Anwenderbeteiligung durch entsprechende Modifikation der Kommunikationsform zwischen Anwender, Moderatoren, Entscheider und Entwickler in Fällen, in denen von den Anwendern konkrete und konstruktive Anwenderbeiträge zu Prototypen erwartet werden, reduziert werden können. Der Ansatz des OpenProposal Annotationssystems stellte sich hierfür als geeignet heraus.

- Die Ergebnisse zu den anderen Methoden decken sich mit denen in der Literatur dokumentieren Phänomen (vgl. [Allen et al., 1993; Andreasen et al., 2007; Baroudi et al., 1986; Bruun et al., 2009; Carrizo et al., 2008; Iivari, 2004; Ives und Olson, 1984; Kujala, 2008; McKeen und Guimaraes, 1997]). Interviews bzw. Usability Tests mit der „Thinking aloud“-Methode und Videoanalyse sind eine sehr effektive Methode zur Anwenderbeteiligung. Sie erfordern allerdings einen hohen Aufwand und produzieren nur wenige konstruktive Anwenderbeiträge. Projekt-Wiki, JIRA und Feedbackformular wurden aufgrund ihrer Komplexität nicht benutzt.

10 Schlussbemerkungen

In diesem Kapitel werden zunächst die wesentlichen Ergebnisse dieser Arbeit zusammengefasst und der erbrachte Forschungsbeitrag aufgezeigt. Des Weiteren werden die Grenzen des Forschungsansatzes umrissen und diskutiert. Der Ausblick greift mögliche Weiterentwicklungen auf und motiviert abschließend offene Fragestellungen, die sich aus den Untersuchungen ergeben haben.

10.1 Zusammenfassung der Arbeit

Diese Arbeit befasste sich mit den Chancen und Risiken der Anwenderbeteiligung in Software-Projekten und demonstrierte die Notwendigkeit der Entwicklung einer neuen Methode zur asynchronen Anwenderbeteiligung. Initial wurde die Arbeitshypothese aufgestellt, dass Methoden zur asynchronen Anwenderbeteiligung Potentiale zur Verbesserung der Effizienz und Effektivität der Anwenderbeteiligung besitzen und tiefergehender Untersuchungen bedürfen.

Hierfür wurde zunächst die Anwenderbeteiligung als ein wesentlicher Erfolgsfaktor für Software-Projekte erläutert, der aktuelle Stand der Forschung zu Anwenderbeteiligung aufgearbeitet und Bewertungskriterien für Methoden zur Anwenderbeteiligung entwickelt. Die Bewertungskriterien orientieren sich an den Auswirkungen, die die Wahl der Methode zur Anwenderbeteiligung auf die Effizienz der Entwicklung, auf die Qualität der Anwenderbeiträge und auf die Belastung der Anwender hat.

Anschließend wurden anhand der Bewertungskriterien Potentiale und Grenzen von Methoden zur synchronen Anwenderbeteiligung bestimmt. Dabei konnte herausgearbeitet werden, dass die Stärken der synchronen Methoden in der Qualität der Anwenderbeiträge und der geringen Belastung der Anwender liegen, dass jedoch auch Schwächen vorwiegend in der Effizienz der Entwicklung existieren. Zudem eignen sich synchrone Methoden nur in einem geringen Maße zur Erfassung von Anwenderbeiträgen im Nutzungskontext.

Im Anschluss erfolgte die Analyse der Potentiale und Grenzen von Methoden zur asynchronen Anwenderbeteiligung anhand der Bewertungskriterien. Die Analyse ergab, dass asynchrone Methoden Stärken in der Effizienz der Entwicklung besitzen. Sie sind zudem für die Analyse des Nutzungskontexts der Anwender geeignet. Für die Anwendung in Software-Projekten sind asynchrone Methoden in dieser Form jedoch nicht geeignet, da sie Schwächen in der Qualität der Anwenderbeiträge und der Belastung der Anwender aufweisen.

Daher wurde die neue Methode OpenProposal konzipiert, welche die Vorteile der asynchronen Anwenderbeteiligung beibehalten und ihre Nachteile beheben soll. Hierfür wurden Gestaltungsziele und Anforderungen im Rahmen von Befragungen und Analysen von Projektdaten erhoben und Lösungskonzepte entwickelt. Dabei wurde als zentrales Unterscheidungsmerkmal zu anderen Methoden der Einsatz eines Annotationssystems zur Annotation von Bildschirmfotos vorgeschlagen, das den Anwender bei der Erstellung seiner Anwenderbeiträge unterstützt und für eine bessere Verständlichkeit seiner Wünsche sorgt. Grundidee des Ansatzes ist es, Anwendern die Möglichkeit zu

geben, Anwenderbeiträge für eine Software direkt auf dem Bildschirm einer laufenden Software-Anwendung zu annotieren und diese als annotiertes Bildschirmfoto an den Entwickler zu kommunizieren. Dieser Idee liegt die Erwartung zugrunde, dass Anwender sich bei der Erstellung ihrer Anwenderbeiträge an grafischen Elementen und vorgegebenen Strukturen orientieren können und ihre Beiträge unmittelbar während der Nutzung der Software abgeben können. Für die Integration dieses Ansatzes wurden ein Prozessmodell entwickelt und die Einsatzmöglichkeiten von OpenProposal in den einzelnen Phasen eines Software-Projekts beschrieben.

Darauf aufbauend wurde die Implementierung des OpenProposal Annotationssystems beschrieben und Systemarchitektur, Datenmodell, Benutzeroberfläche und Bedienkonzept dargelegt. Für die Implementierung wurde eine iterative Vorgehensweise gewählt, bei der regelmäßig Benutzertests durchgeführt und Verbesserungsmöglichkeiten identifiziert und eingearbeitet wurden. So konnten auch die Ergebnisse aus den später vorgestellten Fallstudien eingearbeitet und beispielsweise die Implementierung der automatischen Generierung von Titel und Kurzbeschreibung beschrieben werden.

Die Evaluation von OpenProposal erfolgte in zwei Schritten. Im ersten Schritt wurde die Gebrauchstauglichkeit des Annotationssystems im Rahmen von Usability Tests untersucht. Die Usability Tests ergaben sowohl positive Rückmeldungen als auch zahlreiche Verbesserungsideen. Sie konnten aufzeigen, dass das Annotationssystem eine gute Gebrauchstauglichkeit aufweist und das Konzept der Annotation von Bildschirmfotos von den Anwendern akzeptiert wird. Im zweiten Schritt wurde die Evaluation der OpenProposal Methode mithilfe der Methode der vergleichenden Fallstudien durchgeführt. Hierfür wurden zwei Fallstudien geplant, durchgeführt und fallstudienübergreifend ausgewertet. Die Auswertung der Fallstudien ergab, dass OpenProposal zur Verbesserung der Anwenderbeteiligung beitragen konnte und sich die zugrundeliegenden Konzepte als geeignet erwiesen. So konnte mit OpenProposal eine hohe Effizienz der Entwicklung, eine hohe Qualität der Anwenderbeiträge und eine niedrige Belastung der Anwender erzielt werden. Im Vergleich zu anderen Methoden konnte festgestellt werden, dass die Anwender OpenProposal aufgrund der vereinfachten Erstellung von Anwenderbeiträgen den Vorzug geben. Damit konnte aufgezeigt werden, dass eine asynchrone Anwenderbeteiligung mit OpenProposal akzeptiert wird und erfolgversprechend sein kann. Zudem zeigte sich, dass OpenProposal auch für die Kombination mit Interviews bzw. Usability Tests geeignet ist, so dass OpenProposal auch für eine synchrone Anwenderbeteiligung hilfreich sein kann.

Die eingangs aufgestellten vier Forschungsfragen der Arbeit konnten vollständig beantwort werden. Die erste Frage nach den Bewertungskriterien wird mit Kapitel 2 beantwortet. Die Antwort auf die zweite Frage nach den Potentialen und Grenzen von Methoden zur synchronen Anwenderbeteiligung findet sich in Kapitel 3. Den möglichen Nutzen von Methoden zur asynchronen Anwenderbeteiligung zeigt Kapitel 4 auf. Mit der vierten Frage nach einer neuen Methode zur asynchronen Anwenderbeteiligung, mit der eine hohe Effizienz der Entwicklung, eine hohe Qualität der Anwenderbeiträge und eine niedrige Belastung der Anwender sichergestellt werden kann, befassen sich die Kapitel 5, 6, 7, 8 und 9.

10.2 Hauptbeitrag der Arbeit

Diese Arbeit ist die erste wissenschaftliche Untersuchung zur Eignung von Annotationssystemen zur asynchronen Anwenderbeteiligung in Software-Projekten. Hierfür wurden unterschiedliche Aspekte der Anwenderbeteiligung beleuchtet, um eine effektive und effiziente Anwenderbeteiligung in Software-Projekten zu ermöglichen. Dabei konnten folgende eigene Beiträge herausgearbeitet werden:

- Mithilfe der Literaturrecherche zu Rollen in der Anwenderbeteiligung konnten die Rollen Anwender, Entwickler, Entscheider und Moderator als die relevanten Beteiligten in einer Anwenderbeteiligung in Software-Projekten definiert werden, womit eine einheitliche und vereinfachte Definition der relevanten Rollen gegeben werden konnte. Für jede dieser Rollen wurde der Bezug zu den üblichen Rollen in Vorgehensmodellen hergestellt, Aufgaben, Verantwortlichkeiten und Bedürfnisse zugeordnet

- Aus der Literaturrecherche zu UCD, EM und PD wurden relevante Erfolgsfaktoren der Anwenderbeteiligung herausgearbeitet, die in dieser Übersicht bislang nicht in der Literatur vorlagen. Darauf aufbauend wurden Vor- und Nachteile der Anwenderbeteiligung identifiziert. Im Anschluss wurde der Frage nachgegangen, wie sich anhand der ermittelten Vor- und Nachteile Bewertungskriterien für Methoden zur Anwenderbeteiligung ableiten lassen. Hierfür wurde ein Modell über die Auswirkungen von Methoden zur Anwenderbeteiligung in Software-Projekten entwickelt, das die komplexen Zusammenhänge zwischen der Wahl der Methode und dem Erfolg eines Projektes beschreibt.

- Anhand einer Literaturanalyse wurden Methoden zur synchronen und asynchronen Anwenderbeteiligung identifiziert, die möglichen unterschiedlichen Ausprägungen beschrieben und ihre gemeinsamen Eigenschaften ausgearbeitet. Darauf aufbauend wurden anhand einer Literaturanalyse die Potentiale und Grenzen von Methoden zur synchronen und asynchronen Anwenderbeteiligung gemäß den Bewertungskriterien aus Kapitel 2.9 erfasst.

- Anhand einer Literaturanalyse wurden Anforderungen an Annotationssysteme erarbeitet und ihre unterschiedlichen Einsatzmöglichkeiten zusammengefasst. Darauf aufbauend wurden Vor- und Nachteile des Einsatzes von Annotationssystemen bei einer Anwenderbeteiligung erörtert.

- Für die Konzeption von OpenProposal wurde eine systematische Anforderungsanalyse durchgeführt, die in einer ausführlichen Liste von Anforderungen an Methoden zur asynchronen Anwenderbeteiligung resultierte.

- Darauf aufbauend wurde mit OpenProposal ein Lösungskonzept entwickelt und im Rahmen von Usability Tests und Fallstudien evaluiert. Im Usability Kapitel konnte nachgewiesen werden, dass das OpenProposal Annotationssystem für den Einsatz in realen Software-Projekten geeignet ist. Die Anwender sehen in dem Ansatz eine Arbeitserleichterung und konnten sich schnell und leicht mit dem Bedienkonzept anfreunden. In den Fallstudien konnte gezeigt werden, dass OpenProposal zu einer Verbesserung der Anwenderbeteiligung führen kann. Die positive Bestätigung der Hypothesen zeigt auf, wie mit OpenProposal die Nachteile der existierenden Methoden zur asynchronen Anwenderbeteiligung durch entsprechende Modi-

fikation der Kommunikationsform zwischen Anwender, Moderator, Entscheider und Entwickler in Fällen, in denen von den Anwendern konkrete und konstruktive Anwenderbeiträge zu Prototypen erwartet werden, reduziert werden können. Der Ansatz des OpenProposal Annotationssystems stellte sich hierfür als geeignet heraus

- Im Rahmen der Evaluation konnte gezeigt werden, dass vergleichende Fallstudien für die Evaluation von neuen Methoden zur Anwenderbeteiligung geeignet sein können. Dabei wurde demonstriert, wie Fallstudien geplant und Ergebnisse fallstudienübergreifend ausgewertet werden können. Für die Fallstudien wurden zudem abhängige Variablen definiert, die als Messgrößen zur Bewertung von Methoden zur Anwenderbeteiligung herangezogen werden können. Diese Variablen stellen ausgehend von der Ableitung der Hypothesen und der Forschungsfragen der Fallstudien die Operationialisierung der Bewertungskriterien aus Kapitel 2.9 dar und können für andere Fallstudien angewendet werden.

Die wissenschaftliche Anerkennung der Arbeit konnte durch Publikationen und regen Informationsaustausch mit Experten sichergestellt werden. Der Lösungsansatz von OpenProposal und die Ergebnisse der Untersuchungen wurden auf zahlreichen nationalen und internationalen Konferenzen im Rahmen von Vortragsreihen an Universitäten und bei Software-Unternehmen vorgestellt und diskutiert (vgl. [Lohmann und Rashid, 2008; Maalej et al., 2009; Rashid et al., 2006a; Rashid et al., 2006b; Rashid, 2007; Rashid, 2009; Rashid und Dinther, 2009]). Die Arbeiten in dieser Thematik führten in Zusammenarbeit mit anderen Forschern aus dieser Thematik zu Initiierung einer Veranstaltungsreihe zu Open Design Spaces[38] auf drei Konferenzen[39], bei denen die Vision der für Anwender und Entwickler offenen Gestaltungsräume in Unternehmen und Communities im Vordergrund stehen. Der größte Mehrwert der Veranstaltungen ergab sich dabei durch die Brückenbildung zwischen den unterschiedlichen Forschungsgemeinschaften (u.a. aus den Themenfeldern Usability Engineering, Software Engineering, Requirements Engineering, End-User Development und Participatory Design) untereinander und mit Unternehmen. Die Veranstaltungsreihe fand große Zustimmung und wird fortgesetzt.

10.3 Grenzen des Forschungsansatzes

Die Untersuchungen zu OpenProposal offenbarten konzeptionelle und methodische Grenzen des Forschungsansatzes. Die konzeptionellen Grenzen stellen die eingeschränkten Einsatzmöglichkeiten bei der Annotation von Bildschirmfotos für die Erfassung von Anwenderbeiträgen dar. Die methodischen Grenzen finden sich in den eingeschränkten Möglichkeiten zur Untersuchung von Methoden zur Anwenderbeteiligung.

Die Ergebnisse der Untersuchungen attestieren OpenProposal positive Effekte auf die Durchführung und Nachbereitung der Anwenderbeteiligung. Allerdings wurde von den Teilnehmern der Fallstudien angemerkt, dass sich OpenProposal hauptsächlich für Vorschläge eignet, die sich auf die Bildschirm-

[38] Workshop-Reihe Open Design Spaces, http://www.open-design-spaces.de, abgerufen am 01.03.2010

[39] International Symposium on End-User-Development (IS-EUD 2009), Mensch und Computer 2009 (MuC 2009) und DIS 2010.

oberfläche einer Software beziehen. Zur Erfassung von allgemeinen Anforderungen eignet sich OpenProposal hingegen nicht. Außerdem fiel es den Anwendern schwer, mit OpenProposal Beiträge zu formulieren, die mehrere Dialogfenster gleichzeitig betreffen. Darüber hinaus wurde mehrheitlich eine Kombination von OpenProposal mit anderen etablierten Methoden bevorzugt.

Aus methodischer Sicht stellten die Möglichkeiten der empirischen Software-Technik Forschung eine Schwachstelle der Arbeit dar. Der Zugang zum anwendungsnahen Forschungsfeld in Software-Projekten ist in der Regel aufgrund der knapp bemessenen Ressourcen und dem Zeitdruck der Projekte nur eingeschränkt möglich. So stellte sich die Methode der Fallstudie hinsichtlich der vorhandenen Möglichkeiten als ein angemessenes Instrument dar, mit dem sich die aufgestellten Hypothesen untersuchen ließen. Allerdings hätten sich mit einem uneingeschränkten Zugang weitaus wertvollere Daten erheben lassen, mit denen die in dieser Arbeit erfassten Daten eine höhere Aussagekraft gewonnen hätten. Dies hätte zu einem noch breiteren Fundament für die Überprüfung der Thesen führen können. Ein Experiment, bei dem kein Zugang zum Anwendungsfeld notwendig gewesen wäre, hätte sich aufgrund der Anzahl der nicht kontrollierbaren Variablen und der Verfälschung des Praxisbezugs als nicht sinnvoll erwiesen.

Eine weitere Schwachstelle des Forschungsansatzes stellen die ungenauen Merkmalsausprägungen der Bewertungskriterien und die teilweise ungenaue Definitionen der Anforderungen an das Annotationssystem dar. Anstelle von überprüfbaren quantitativen Merkmalen konnte lediglich eine vereinfachte Ordinalskala („Niedrig“ < „Mittel“ < „Hoch“) zur Veranschaulichung herangezogen werden. Dies ist dem Umstand geschuldet, dass in der Literatur keine exakten und einheitlichen Basisdaten für die quantitative Beurteilung des Aufwandes der Entwicklung, der Qualität der Anwenderbeiträge und der Belastung der Anwender existieren. Für Software-Projekte existieren aufgrund der komplexen Rahmenbedingungen im Allgemeinen keine einheitlichen fest vorgegebenen Messgrößen. Für jedes Software-Unternehmen, für jedes Software-Projekt und für jede Software finden sich unterschiedliche Ausgangslagen und Zielvorgaben, die sich im Laufe der Zeit auch ändern. Daher ist davon auszugehen, dass die Ermittlung dieser Basisdaten auch in Zukunft nicht möglich ist. Vielmehr eignen sich relative Messgrößen in Form von Ordinalskalen als Hilfsmittel, um eine grundlegende Bewertung durchführen zu können.

10.4 Technische Weiterentwicklung

Im Laufe der Untersuchungen entstanden zahlreiche neue und weiterführende Fragestellungen, die vor allem Optionen zur Weiterentwicklungen des OpenProposal Annotationssystems adressieren. Im Folgenden werden potentielle Anschlussarbeiten erläutert. Diese betreffen die Verwaltung und Bewertung von Anwenderbeiträgen, die Konfigurierbarkeit und Individualisierbarkeit von OpenProposal, die Kontexterkennung, die Titelgenerierung und die pro-aktive Motivation der Anwender im laufenden Betrieb sowie weitere geeignete Anwendungsszenarien.

10.4.1 Verwaltung und Bewertung von Anwenderbeiträgen

Bei der Konzeption von OpenProposal wurde bewusst eine eigene Lösung zur Verwaltung der Anwenderbeiträge vermieden. Stattdessen wurde die Möglichkeit einer Anbindung über eine Kommunikationsschnittstelle geschaffen, mit der die annotierten Bildschirmfotos in einem lokalen Ordner

gespeichert, per Mail versendet oder als Issue in einem Issue-Tracker gespeichert werden können. Als Folge dieser Gestaltungsentscheidung musste in Kauf genommen werden, dass die Verwaltung der Anwenderbeiträge nicht direkt in OpenProposal enthalten ist, sondern in Fremdkomponenten ausgelagert wird und somit kein Einfluss darauf ausgeübt werden kann. Die Moderatoren in den Fallstudien äußerten jedoch hierzu Verbesserungsvorschläge, um die Anwenderbeiträge übersichtlicher darzustellen. Zudem wurde in Erwägung gezogen, Anwendern zukünftig die Möglichkeit zu geben die abgegebenen Anwenderbeiträge gegenseitig zu bewerten, um die Prioritäten der Beiträge besser einschätzen zu können. Aus diesem Grund wurde neben dem Annotationsmodus versuchsweise ein Bewertungsmodus in OpenProposal implementiert und Anwendern zum Testen bereitgestellt.

Abbildung 10.1 illustriert die Funktionsweise des Bewertungsmodus von OpenProposal. Über das Hauptmenu wird zwischen Annotations- und Bewertungsmodus gewechselt. Zu Beginn kann der Anwender die Software bzw. das Software-Projekt auswählen und sich anschließend alle offenen Vorschläge zu der Software auflisten lassen. Zu jedem Vorschlag werden das annotierte Bildschirmfoto im Vollbild, der Titel, die Kurzbeschreibung, der Bearbeitungszustand und die evtl. bereits abgegebenen Bewertungen anderer Anwender angezeigt. Auf einer „Fünf-Sterne“-Skala kann der Grad der Zustimmung abgegeben und bei Bedarf um einen textlichen Kommentar ergänzt werden. Jede Bewertungsabgabe wird eindeutig dem Benutzernamen zugeordnet, so dass von jedem Benutzernamen nur eine einzige Bewertung abgegeben werden kann. Falls der Anwender dies wünscht, kann er seine Bewertung nachträglich ändern.

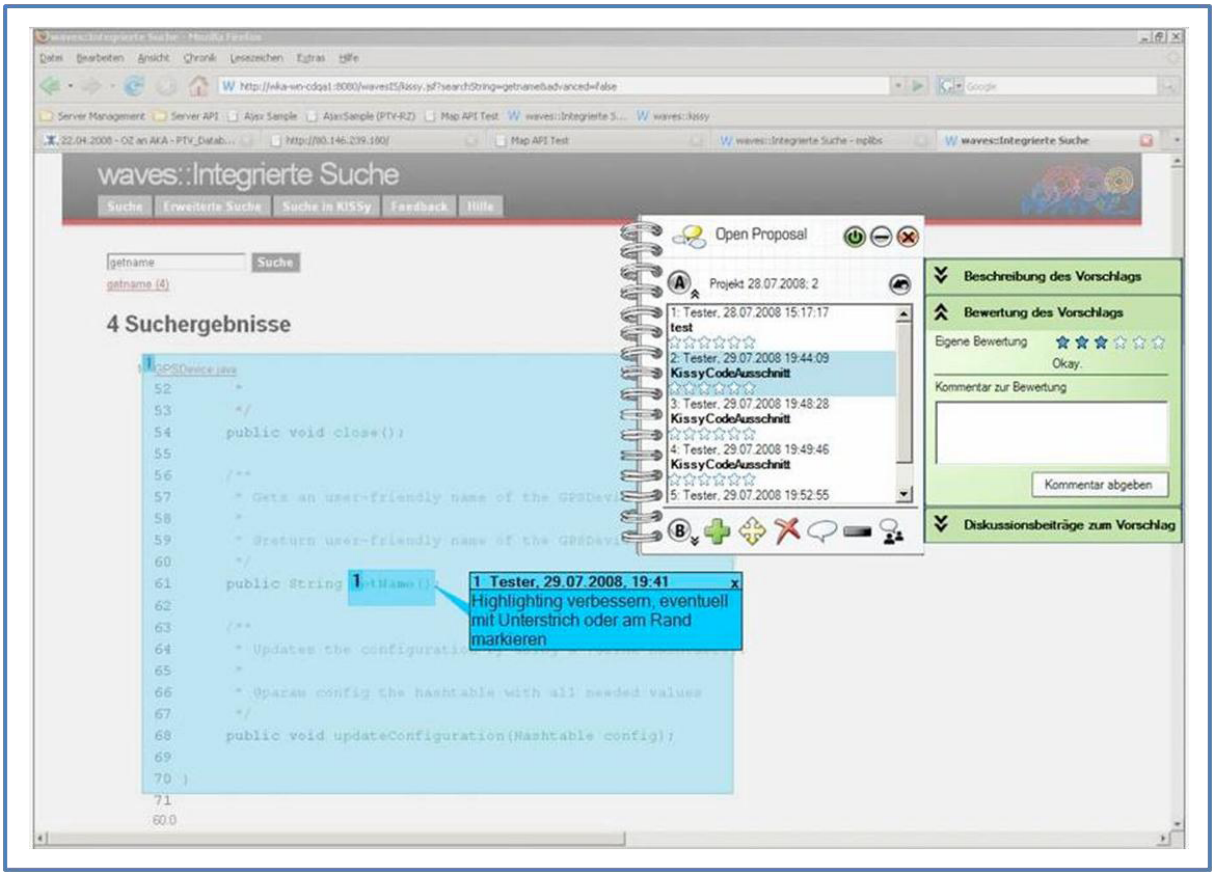

Abbildung 10.1: Beispielhafte Funktionsweise des Bewertungsmodus von OpenProposal

Die ersten Tests mit Anwendern und Entscheidern im Anschluss an die WAVES-Fallstudie waren vielversprechend. Für die Anwender stellte die Bewertung einen geringen Aufwand dar. Die Entscheider fanden diese Funktion sehr hilfreich für ihre Arbeit. Allerdings ergaben sich auch Probleme in der Übersichtlichkeit der Darstellung. Daher ist dieser Ansatz in weiteren Arbeiten ausführlich zu evaluieren und an die Wünsche der Anwender und Entscheider anzupassen.

10.4.2 Individualisierbarkeit von OpenProposal

Eine weitere offene Fragestellung stellt die Anpassbarkeit und Individualisierbarkeit von OpenProposal dar. In Gesprächen mit interessierten Unternehmen wurde häufig der Wunsch geäußert, OpenProposal an die Gegebenheiten eines Unternehmens anzupassen. Beispielsweise sollten abhängig von der Zusammenstellung der Anwender und der getesteten Software bestimmte Bereiche ausgeblendet werden. Für die weitere Entwicklung von OpenProposal bedeutet dies, den Bedarf an Flexibilität zu ermitteln und in der Implementierung zu berücksichtigen. Eine erste Maßnahme wäre hierbei, in der Konfiguration einstellen zu können, welche Annotationswerkzeuge angezeigt werden und ob das Hauptmenü deaktiviert sein soll.

10.4.3 Automatische Generierung von Titel und Kurzbeschreibung

Die automatische Generierung von Titel und Kurzbeschreibung zu den annotierten Bildschirmfotos wurde erst nach der zweiten Fallstudie implementiert. Die Grundlage der Umsetzung wurde den Vorschlägen der Entscheider und Entwickler aus der zweiten Fallstudie entnommen. Allerdings ist diese Komponente noch zu verfeinern und bestmöglich an die Anforderungen der Entscheider und Entwickler anzupassen. Während die Daten aus den Systemparametern noch sehr stark strukturiert verarbeitet werden können, sind die ergänzenden Textbeschreibungen zu den Annotationen vollständig unstrukturierter Natur. Hier können Methoden der Künstlichen Intelligenz hilfreich sein, um mit Hilfe von Text Mining, Textanalyse bzw. Texterkennung mit sprachnatürlichen Texten eine Lösung für aussagekräftige Titel und Kurzbeschreibungen zu entwickeln. Allerdings ist bei Texten mit ca. 15 Wörtern pro Annotation aufgrund der geringen Textlänge mit großen Schwierigkeiten zu rechnen. Erschwerend kommt hinzu, dass im Text Abkürzungen und unvollständige Sätze enthalten sind, deren Inhalte auch von den Entscheidern und Entwicklern möglicherweise nicht nachvollzogen werden konnten.

10.4.4 Kontexterfassung und pro-aktive Motivation

Die Potentiale und fortführende Weiterentwicklungsmöglichkeiten der Erfassung von Anwenderbeiträgen im Nutzungskontext wurde unter dem Arbeitstitel „When Users Become Collaborators: Towards Continuous and Context-Aware User Input“ von Maalej, Happel und Rashid [2009] erarbeitet und als Positionspapier publiziert. Der Ansatz basiert auf dem Konzept, dass die Anwender kontinuierlich und kontext-sensitiv in die Entwicklung von Software einbezogen werden sollen, um deren Wünsche und Probleme im Betriebsalltag besser verstehen und die Software an die tatsächlichen Bedürfnisse anpassen zu können. Dabei bilden drei Bausteine das Fundament der angedachten Lösungsarchitektur:

- Kontexterfassung und -interpretation: Der Kontext der Anwender könnte durch eine integrierte Protokollierung der Nutzungsdaten aufgenommen und darüber Rückschlüsse auf Anwenderverhalten und -bedürfnisse gezogen werden. Hierfür könnten Komponenten zur Er-

fassung und Interpretation der Kontextdaten zum Einsatz kommen. Dies könnte darüber Aufschluss geben, welche Funktionalität einer Software noch zu verbessern ist, wenn Anwender diese beispielsweise nie nutzen oder länger als notwendig mit deren Bedienung beschäftigt sind.

- Grafische Annotation: Die Funktionalität des OpenProposal Annotationssystems könnte Teil dieser Lösung sein und dem Anwender die Erstellung von Anwenderbeiträgen erleichtern. Dabei ließe sich das Konzept der Annotation von Bildschirmfotos auch direkt in Software-Anwendungen integrieren. Die Werkzeugleiste zur Annotation von Bildschirmfotos würde somit direkt aus der Anwendung verwendbar und erfordert nicht mehr das zusätzliche Starten des OpenProposal Annotationssystems.
- Pro-aktive Assistenz: Durch die Beobachtung des Anwenderverhaltens könnten diese Information zur pro-aktiven Assistenz des Anwenders bei der Benutzung von Software eingesetzt werden. Beispielsweise könnte Anwendern bei auffällig ineffizienter Bedienung aufgezeigt werden, wie andere Anwender dies besser gelöst haben. Diese Informationen könnte auch dazu genutzt werden, die in einer Software hinterlegten Workflows an das Anwenderverhalten anzupassen.

Dieser Vision stehen zahlreiche Herausforderungen und Fragestellungen u.a. zu technischer Machbarkeit, Nutzen, Datenschutz, Aufwand und Angemessenheit gegenüber, die in zukünftigen Arbeiten zu untersuchen bzw. bereits in den Forschungsprojekten TEAM[40] und GlobaliSE[41] in Bearbeitung sind.

Ein weiteres zu untersuchendes Szenario wäre, den Anwender während der Nutzung der Software pro-aktiv um eine kurze Rückmeldung zu bitten. In diesem Szenario werden ausgewählte Bereiche der Software vom Entwickler markiert, zu denen Rückmeldungen des Anwenders gewünscht wird. Der Anwender verwendet die Software im täglichen Betrieb und wird bei den vom Entwickler markierten Bereichen zu einem Kommentar gebeten. Als Kommentar könnten ein Text, eine Bewertung auf einer Fünf-Sterne-Skala oder eine grafische Annotation abgegeben werden. Technisch wäre dies möglich, wenn OpenProposal ständig als Hintergrundprozess gestartet ist, über die Kontexterkennung erkennt, welche Bereiche der Software geöffnet sind und bei Übereinstimmung eine grafische Markierung an der entsprechenden Stelle einblendet. Zweck dieser Funktionalität wäre es, dass der Anwender selbst nicht aktiv an die Abgabe von Vorschlägen denken muss sondern pro-aktiv um eine Kommentierung gebeten wird. Zudem kann der Entwickler festlegen, zu welchen Komponenten Feedback erwünscht ist. Die technische Umsetzung stellt eine große Herausforderung dar, da die Kontexterkennung vom Umfang und Güte der Systemparameter der Software abhängt, auf die OpenProposal zugreifen kann. Außerdem ist die grafische Positionierung von der Bildschirmauflösung und Fenstereinstellung abhängig. Eine technisch machbare Lösung könnte sein, dass in der Software eine Richtlinie zur Hinterlegung von Systemparametern berücksichtigt wird, so dass OpenProposal die markierten Bereiche der Software eindeutig erkennen und zuordnen kann. Allerdings ist auch dieses Szenario einer gründlichen Evaluation zu unterziehen, um die Akzeptanz der Anwender zu

[40] Projekt TEAM, http://www.team-project.eu/, abgerufen am 01.03.2010

[41] Projekt GlobaliSE, http://www.globalise-projekt.de, abgerufen am 01.03.2010

beobachten. Im Vordergrund der Untersuchungen sollte die Frage stehen, ob die Anwender diese Form der Aufforderung wie bei den automatischen Problemberichten ignorieren oder aufgrund der zielgerichteten und transparenten Darstellungen ernst nehmen.

10.4.5 Weitere Anwendungsszenarien

Die Übertragbarkeit von OpenProposal auf andere grafischen Repräsentationen von Inhalten abseits von Software-Produkten stellt zusätzliche weiterführende Fragestellung dar. Dabei steht nicht die Anwenderbeteiligung sondern die kollaborative Verbesserung von grafischen Darstellungen im Vordergrund. Hierbei kann OpenProposal für die Abgabe von Rückmeldungen zu Diagrammen bzw. Grafiken aller Art verwendet werden, beispielsweise zur Annotation von Grafiken in einem Textdokument. Dies kann z.B. für Korrekturzyklen hilfreich sein. Moderne Textverarbeitungsprogramme wie z.B. Microsoft Word bieten Funktionen zur Kommentierung von Textstellen an. Allerdings mangelt es an Funktionen zur Annotation von Grafiken, die im Textdokument eingebettet sind.

Ein anderes Anwendungsszenario stellen Prozessmodelle dar. Die Konzepte von OpenProposal können in einem Prozessmodellierungswerkzeug in Form von Kommentarwerkzeugen integriert und für eine kollaborative Entwicklung von Prozessmodellen verwendet werden. Solange die Hersteller dieser Programme eine solche integrierte Werkzeugleiste nicht anbieten, könnte OpenProposal in der aktuellen Form verwendet werden, und die annotierten Bildschirmfotos können per Email an den Autor der Prozessmodelle gesendet werden. Allerdings sollte eine integrierte Umsetzung bevorzugt werden, weil damit direkt auf die Objekte im Prozessmodell Bezug genommen und auf die Erstellung von Bildschirmfotos verzichtet werden kann. Dies würde die weitere Bearbeitung der Prozessmodelle erleichtern, da die Kommentare in den Prozessmodellen verwaltet werden könnten.

Bei einem Unternehmen der Fallstudie WAVES wurde zudem vorgeschlagen, OpenProposal auch für die technische Dokumentation zu verwenden. Grundüberlegung war, dass für die Erstellung der technischen Dokumentation zu Software-Produkten häufig auf die Bildschirmoberfläche referenziert wird. OpenProposal kann möglicherweise diesen Prozess beschleunigen, da Erstellung und die strukturierte Annotation der Bildschirmfotos in einem Schritt erfolgen kann. Für eine bestmögliche Unterstützung müssten allerdings die Klassen der Annotationswerkzeuge und das Absenden der annotierten Bildschirmfotos an das Szenario angepasst werden.

10.5 Ausblick

Diese Arbeit bildet einen ersten Grundstein für weitere Untersuchungen von Methoden zur asynchronen Anwenderbeteiligung. Im ersten Schritt konnte beschrieben werden, wie solche Methoden geschaffen sein müssen und wie sie in Software-Projekte integriert werden können. Dabei eröffnet diese Arbeit neue bzw. tiefer gehende Fragestellungen, die für Nachfolgearbeiten geeignet sein können.

In erster Linie kann die Evaluation von OpenProposal auf weitere Software-Projekte ausgeweitet und die bisherigen Ergebnisse überprüft werden. Dabei können auch zahlreiche andere Methoden zum Vergleich hinzugezogen werden, so dass schließlich eine Gegenüberstellung aller Methoden möglich wird. Zudem können die Bewertungskriterien an Methoden zur Anwenderbeteiligung im Rahmen einer Umfrage verifiziert und durch Gewichtungen ergänzt werden. Damit kann der Zusammenhang

zwischen Bewertungskriterien und Projekterfolg quantifiziert werden. Dies führt schließlich zu einer Entscheidungsunterstützung bei der Wahl der Methode zur Anwenderbeteiligung vor Beginn eines Software-Projektes. Eine Vorlage stellen die Arbeiten von Carrizo et al. [2008], Davis et al. [2006] und Dieste und Juristo [2010] dar, die eine ähnliche Richtung eingeschlagen haben.

Interessant wäre zudem die Untersuchung der Effekte der Wahl der Methode zur Anwenderbeteiligung auf die Demokratisierung von Software-Projekten. Dank einer verbesserten Kommunikation zwischen Anwender, Moderator, Entscheider und Entwickler könnte die Anwenderbeteiligung zunehmen und Anwender stärker in die Entscheidungsprozesse einbezogen werden. Dies hätte starken Einfluss auf die bisherige Praxis der Entscheidungsfindung, bei der eine Zunahme der Kommunikation und Abstimmung zwischen den Beteiligten zu erwarten ist. Hierfür sind weitere asynchrone Methoden zu entwickeln bzw. OpenProposal zu erweitern, um diesen Prozess von der Erfassung von Anwenderbeiträgen bis zum Beschluss von Anforderungen zu unterstützen. Denkbar ist hierfür ein Bewertungssystem, mit dem Anwender ihre Einschätzung zu den gesammelten Anwenderbeiträgen und dem aktuellen Stand der Anforderungen abgeben können. Für die Anwendung von asynchronen Methoden müsste geprüft werden, ob die Bewertung der Anwender hilfreich ist und ob die Kommunikation mit den Anwendern reibungslos verläuft. Außerdem ist zu untersuchen, ob durch diese Form der Demokratisierung positive Effekte auf die Qualität des Systems und die Zufriedenheit der Anwender erzielt werden können.

Eine andere Richtung könnte mit dem Ansatz von Fischer et al. [2004] mit Meta Design eingeschlagen werden. Dabei wird die These vertreten, dass Software in Zukunft nicht auf eine einzelne Funktionalität beschränkt sondern mit einer generischen Konfiguration ausgestattet wird, so dass der Anwender aufbauend auf einer „Meta-Software“ schließlich seine eigenen Anwendungen zusammenstellen kann. Für Methoden zur asynchronen Anwenderbeteiligung stellt sich hierbei die Frage, wie Anforderungen an solche generischen Komponenten erhoben werden können. OpenProposal könnte hierfür einen interessanten Ansatz darstellen, der allerdings für diese Art von Software untersucht werden müsste.

Ein weiterer Ansatz für zukünftige Arbeiten stellt die Untersuchung von asynchroner Anwenderbeteiligung bei mobilen Anwendungen dar. Der Einsatz von mobilen Anwendungen hat in den letzten Jahren stark zugenommen und wird aufgrund der zunehmenden Miniaturisierung und Leistungsfähigkeit sowie innovativer Bedienkonzepte und Software noch weiter zunehmen. In ersten Arbeiten wurde beispielhaft untersucht, wie eine mobile Anwendung auf einem mobilen Gerät mit OpenProposal getestet werden kann. Abbildung 10.2 zeigt den Versuchsaufbau. Auf dem mobilen Gerät wurde eine Schwenkhalskamera montiert, so dass die Anwendung auf dem mobilen Gerät mithilfe einer Kamera aufgenommen und auf einem Computer angezeigt wird. Der Anwender testet das mobile Gerät und annotiert seinen Anwenderbeitrag mithilfe des Computers auf dem Bild der Kamera.

Erste Untersuchungen mit 20 Testpersonen zeigten, dass Anwender dieser Form der Erstellung von Anwenderbeiträgen positiv gegenüberstehen. Im Vergleich zu Interviews bzw. Usability Tests mit der „thinking aloud“-Methode wurden mit OpenProposal keine zusätzlichen Beiträge erstellt. Für die Entwickler waren die annotierten Bildschirmfotos von großer Hilfe, da sie auf diese Weise den Kontext des Anwenders auf einen Blick erfassen konnten ohne dafür das vollständige Video des Usability Tests ansehen zu müssen.

OpenProposal könnte zudem in Open-Source-Software-Projekten Anwendung finden. Die Ansprüche an die Gebrauchstauglichkeit von Open-Source-Software steigt zunehmend und ist bereits Gegenstand mehrerer wissenschaftlicher Untersuchungen (vgl. [Andreasen et al., 2006; Raza et al., 2010]). Da die Kommunikation in Open-Source-Software-Projekten vorwiegend asynchron verläuft, könnte OpenProposal vor allem im Bezug auf die Gestaltung der grafischen Benutzeroberfläche hilfreich für die Kommunikation zwischen Entwickler und Anwender sein. Allerdings beruht die Arbeit in Open-Source-Software-Projekten häufig auf einer ideellen Motivation der Entwickler und Anwender, so dass die Problembereiche der Anwenderbeteiligung anders zu deuten sind und die Effizienz der Entwicklung und die Belastung der Anwender sich anders als in klassischen Software-Projekten verhalten können.

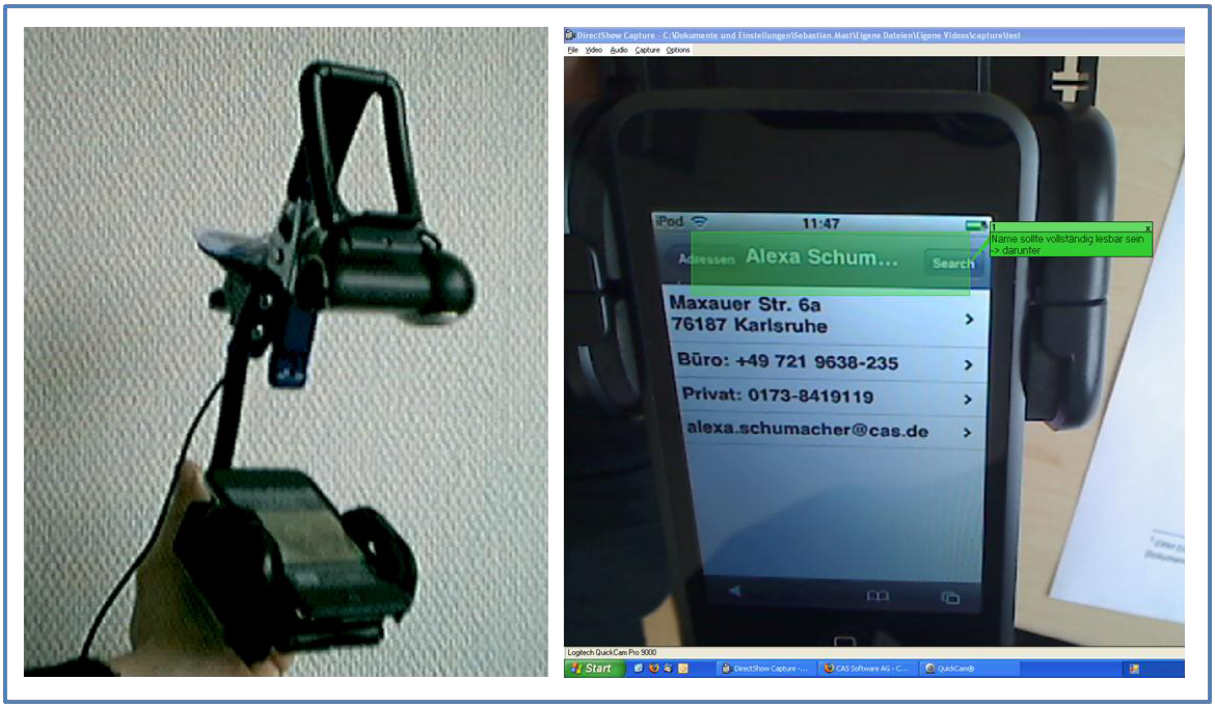

Abbildung 10.2: OpenProposal auf mobile Geräte

In Internetseiten, webbasierte Anwendungen und Web 2.0 Communities lassen sich Methoden zur asynchronen Anwenderbeteiligung leicht integrieren. Anwender können mithilfe eingebauter Kommentar- und Bewertungsfunktionen sowie speziellen Anwendungen wie z.B. User Voice[42], Usabilla[43] und Kampyle[44] Kommentare abgeben. Diese Ansätze sind vergleichbar mit dem integrierten Feedbackformular aus der Fallstudie 2 und es ist zu erwarten, dass sie für einzelne Software-Projekte nicht geeignet sind. Allerdings können sie bei einer großen Zahl an Anwendern, wie es bei Internetseiten bzw. webbasierten Anwendungen üblicherweise gegeben ist, hilfreich sein. Interessant wäre es, zu untersuchen, welche Vor- und Nachteile die bisherigen Ansätze besitzen und ob OpenProposal mit dem Ansatz der Annotation von Bildschirmfotos Verbesserungen herbeiführen kann.

[42] uservoice, http://uservoice.com, abgerufen am 17.03.2009

[43] Usabilla, http://www.usabilla.com, abgerufen am 17.03.2009

[44] Kampyle, http://www.kampyle.com, abgerufen am 01.05.2010

Eine offene Frage bleibt zudem diejenige nach den Effekten von finanziellen Anreizen zur Steigerung der Anwenderbeteiligung, wie sie Mayhew [1999] und Shneiderman [2002] postulieren. Für das betriebliche Vorschlagswesen sind solche Anreizsysteme bereits etabliert. Bei der Anwenderbeteiligung in Software-Projekten wird für externe Anwender üblicherweise ein kleiner Beitrag für die Unkosten erstattet. Dies erfolgt aber nicht abhängig von der Anzahl oder der Qualität der Anwenderbeiträge. Interessant wäre an dieser Stelle, ob mit einem leistungsbezogenes Anreizsystem eine höhere Qualität der Anwenderbeiträge herbeigeführt werden kann und wie ein solches Anreizsystem zu gestalten ist. Dies könnte vor allem bei einer asynchronen Anwenderbeteiligung interessant sein, wenn sich Anwender nebenberuflich engagieren können. Ein Vorbild für ein solches System könnte das System Mechanical Turk[45] darstellen, bei dem Menschen für die Erledigung von webbasierten Aufgaben ein Entgelt abhängig von der Qualität und Anzahl der erbrachten Leistungen erhalten. Eine Bewertung dieses Ansatzes führten Kittur et al. [2008] durch, die feststellten, dass für diese Form der Anwenderbeteiligung die Aufgabenstellung sorgfältig erstellt werden muss, da Mechanical Turk ausschließlich kleine Aufgabenstellungen zulässt und nicht für ausführliche Tests von Software geeignet ist. OpenProposal könnte hier für die Kommunikation hilfreich sein, um die Erstellung von Anwenderbeiträgen zu vereinfachen und die Qualität der Anwenderbeiträge erhöhen kann. Allerdings bedarf es hierfür weiterer ausführlicher Untersuchungen.

Schlussendlich zeigt der in dieser Arbeit entwickelte Forschungsansatz auf, dass Methoden zur asynchronen Anwenderbeteiligung, wie z.B. OpenProposal, Potentiale zur Verbesserung der Anwenderbeteiligung in Software-Projekten besitzen. Dabei spielt vor allem die Kommunikationsform zwischen Anwendern und Entwicklern eine bedeutende Rolle. In den Untersuchungen hat sich aber auch gezeigt, dass die Fragestellungen zur Anwenderbeteiligung in Software-Projekten noch längst nicht ausgeschöpft sind - trotz der langen Historie der bisherigen Untersuchungen unterschiedlicher Forschungsgemeinschaften. Im Zuge der rasant fortschreitenden Akzeptanz der Computer- und Internetnutzung in der breiten Bevölkerung wird das Themenfeld der Anwenderbeteiligung weiterhin eine besondere Bedeutung besitzen, an der sich Unternehmen in ihrer Kundennähe messen lassen müssen.

[45] Mechanical Turk, http://www.mturk.com, abgerufen am 01.05.2010

Literaturverzeichnis

[Aaker und Day, 2003] Aaker, D. und Day, G.S. *Marketing Research.* Wiley, New York, USA, 2003.

[Abran und Moore, 2004] Abran, A. und Moore, J.D. *Guide to the Software Engineering Body of Knowledge (SWEBOK) 2004 Version*. IEEE Computer Society, 2004.

[Adler und van Doren, 1972] Adler, M.J. und van Doren, C. *How to Read a Book.* Simon & Schuster, New York, USA, 1972.

[Agresti, 1986] Agresti, W.W. *The conventional software life-cycle model: its evolution and assumptions.* In: *New paradigms for software development,* Agresti, W.W. (Hrsg). IEEE Computer Societxy Press, Washington DC, USA, 1986, S. 2-5.

[Alexander, 1999] Alexander, I. *Is There Such a Thing as a User Requirement?* Requirements Engineering, 4, 1999, S. 221-223.

[Alexander und Maiden, 2004] Alexander, I. und Maiden, N. *Scenarios, stories, use cases: through the system development life-cycle.* Wiley, Chichester, GBR, 2004.

[Alford, 1977] Alford, M.W. *A requirements engineering methodology for teal time processing requirements*. IEEE Transactions on Software Engineering, SE-3 (1), 1977, S. 60-69.

[Allen et al., 1993] Allen, C.D., Ballman, D., Begg, V., Miller-Jacobs, H.H., Muller, M., Nielsen, J. und Spool, J. *User Involvement In The Design Process: Why, When & How?* INTERCHI, 1993.

[Andreasen et al., 2006] Andreasen, M.S., Nielsen, H.V., Schrøder, S.O. und Stage, J. *Usability in Open Source Software Development: Opinions and Practice*. Information Technology and Control, 35 (3), 2006, S. 303-312.

[Andreasen et al., 2007] Andreasen, M.S., Nielsen, H.V., Schrøder, S.O. und Stage, J. *What Happened to Remote Usability Testing? An Empirical Study of Three Methods*. CHI 2007, ACM, San Jose, CA, USA, 2007.

[Andrzejczak und Liu, 2010] Andrzejczak, C. und Liu, D. The effect of testing location on usability testing performance, participant stress levels, and subjective testing experience. Journal of Systems and Software, 83 (7), 2010, S. 1258-1266.

[Apel et al., 1998] Apel, H., Dernbach, D., Ködelpeter, T. und Weinbrenner, P. *Wege zur Zukunftsfähigkeit - ein Methodenhandbuch.* Stiftung Mitarbeit, Bonn, BRD, 1998.

[Atteslander, 2003] Atteslander, P. *Methoden der empirischen Sozialforschung.* 10. Auflage, Springer, Berlin, BRD, 2003.

[Baker et al., 2007] Baker, R.M.S., Kiris, E. und Vasnaik, O. *Testing remote users: an innovative technology*. 2nd International Conference on Usability and Internationalization, Springer, Beijing, CHN, 2007, S. 235-242.

[Balzert, 2000a] Balzert, H. *Lehrbuch der Software-Technik - Bd.1. Software-Entwicklung.* Spektrum - Akademischer Verlag, Heidelberg, Berlin, 2000a.

[Balzert, 2000b] Balzert, H. Lehrbuch der Software-Technik - Bd.2. Software-Management, Software-Qualitätssicherung und Unternehmensmodellierung. Spektrum Akademischer Verlag, Heidelberg, Berlin, BRD, 2000b.

[Barki und Hartwick, 1991] Barki, H. und Hartwick, J. *User participation and user involvement in information system development*. 24. Annual Hawaii International Conference on System Sciences, Hawaii, USA, 1991, S. 487-492.

[Barki und Hartwick, 1994] Barki, H. und Hartwick, J. *Measuring user participation, user involvement, and user attitude*. MIS Quarterly, 18 (1), 1994, S. 60-82.

[Baroudi et al., 1986] Baroudi, J.J., Olson, M.H. und Ives, B. *An Empirical Study of the Impact of User Involvement on System Usage and Information Satisfaction*. Communications of the ACM, 29 (3), 1986, S. 232-238.

[Beck, 1996] Beck, E. *P is for Political*. Participatory Design Conference, Computer Professionals for Social Responsibility, Cambridge, MA, USA, 1996, S. 117–125.

[Beck, 1999] Beck, K. *Extreme Programming Explained: Embrace Change.* Addison-Wesley Longman, Amsterdam, NL, 1999.

[Bell et al., 1977] Bell, T.E., Bixler, D.C. und Dyer, M.E. *An extendable approach to computer aided software requirements engineering*. IEEE Transactions on Software Engineering, SE-3 (1), 1977, S. 49-60.

[Bennington, 1956] Bennington, H.D. *Production of Large Computer Programs*. ONR Symposium on Advanced Programming Methods for Digital Computers, 1956, S. 15-27.

[Bettenburg et al., 2008] Bettenburg, N., Just, S. und Schröter, A. *What makes a good bug report?* SIGSOFT 2008, ACM, Atlanta, Georgia, USA, 2008.

[Bjerknes und Bratteteig, 1995] Bjerknes, G. und Bratteteig, T. *User Participation and Democracy: A Discussion of Scandinavian Research on System Development*. Scandinavian Journal of Information Systems, 7 (1), 1995, S. 73-98.

[Blomberg et al., 1993] Blomberg, J., Giacomi, J., Mosher, A. und Swenton-Wall, P. *Ethnographic Field Methods and Their Relation to Design.* In: *Participatory Design: Principles and Practices,* Dchuler, D. und Namioka, A. (Hrsg). Erlbaum, New Jersey, USA, 1993.

[Blomberg et al., 1997] Blomberg, J., Suchman, L. und Trigg, R. *Back to Work: Renewing Old Agendas for Change.* In: *Computers and Design in Context,* Kyng, M. und Mathiassen, L. (Hrsg). MIT Press, Cambridge, MA, USA, 1997, S. 201–238.

[Bly, 1997] Bly, S. *Field work: is it product work*. Interactions, 4 (1), 1997, S. 25–30.

[Bødker et al., 1987] Bødker, S., Ehn, P., Kammersgaard, J., Kyng, M. und Y., S. A UTOPIAN experience: on design of powerful computer-based tools for skilled graphic workers. In: Computers and democracy: a Scandinavian challenge, Bjerknes, G., Ehn, P. und Kyng, M. (Hrsg). Avebury, Aldershot, 1987, S. 251-278.

[Bødker, 1996] Bødker, S. Creating Conditions for Participation: Conflicts and Ressources in Systems Development. Human-Computer Interaction, 11 (3), 1996, S. 215–236.

[Boehm, 1988] Boehm, B.W. *A Spiral Model of Software Development and Enhancement*. IEEE Computer, May 1988, 1988, S. 61-72.

[Böhler, 2004] Böhler, H. *Marktforschung.* 3. Auflage, Kohlhammer, Stuttgart , BRD, 2004.

[Bohnsack und Przyborski, 2007] Bohnsack, R. und Przyborski, A. *Gruppendiskussionsverfahren und Focus Groups* In: *Qualitative Marktforschung*, Gabler, Wiesbaden, BRD, 2007, S. 491-506.

[Booch, 1994] Booch, G. *Object-oriented Analysis and Design with Applications.* Benhamin Cummings, Redwood City, CA, USA, 1994.

[Booch et al., 1999] Booch, G., Rumbaugh, J. und Jacobson, I. *Das UML-Benutzerhandbuch.* Addison-Wesley, 1999.

[Booch et al., 2007] Booch, G., Maksimchuk, R., Engle, M., Young, B., Conallen, J. und Houston, K. *Object-oriented analysis and design with applications.* 3. Auflage, Addison-Wesley Professional, 2007.

[Borchhardt und Göthlich, 2007] Borchhardt, A. und Göthlich, S.E. *Erkenntnisgewinnung durch Fallstudien.* In: *Methodik der empirischen Forschung,* Albers, S., Klapper, D., Konradt, U., Walter, A. und Wolf, J. (Hrsg). Gabler, Wiesbaden, BRD, 2007, S. 33-47.

[Bortz und Döring, 2003] Bortz, J. und Döring, N. *Forschungsmethoden und Evaluation.* 3. Auflage, Springer, Berlin, BRD, 2003.

[Bottoni et al., 2003] Bottoni, P., Levialdi, S. und Rizzo, P. *An Analysis and Case Study of Digital Annotation.* Springer-Verlag Berlin Heidelberg (Germany), 2003, S. 216-231.

[Bottoni et al., 2004] Bottoni, P., Civica, R., Levialdi, S., Orso, L., Panizzi, E. und Trinchese, R. *MADCOW: a multimedia digital annotation system*. Proceedings of the working conference on Advanced visual interfaces, ACM, Gallipoli, Italy, 2004.

[Brau und Schulze, 2004] Brau, H. und Schulze, H. Minimierung von Reaktanz gegen IT-Einführungsprojekte an industriellen Arbeitsplätzen durch Nutzerpartizipation. Düsseldorf, BRD, 2004, S. 237-244.

[Brockhaus, 2003] Brockhaus, F.A. *Die Brockhaus Enzyklopädie* Brockhaus 2003.

[Brosius et al., 2009] Brosius, H.-B., Koschel, F. und Haas, A. *Methoden der empirischen Kommunikationsforschung.* 5. Auflage, VS Verlag für Sozialwissenschaften, Wiesbaden, BRD, 2009.

[Broy und Rausch, 2005] Broy, M. und Rausch, A. *Das neue V-Modell XT - Ein anpassbares Modell für Software und System Engineering*. Informatik-Spektrum, 28 (3), 2005, S. 220-229.

[Bruegge et al., 2004] Bruegge, B., Creighton, O. und Purvis, M. *Software Cinema.* Wien, AUT, 2004.

[Brush et al., 2002] Brush, A., Bargeron, D., Grudin, J., Borning, A. und Gupta, A. *Supporting Interaction Outside of Class*. CSCL '02, Boulder, Colorado, USA, 2002, S. 425-434.

[Brush et al., 2001] Brush, A.J.B., David, B., Anoop, G. und Cadiz, J.J. *Robust annotation positioning in digital documents*. CHI 2001, ACM Press, 2001, S. 285-292.

[Bruun et al., 2009] Bruun, A., Gull, P., Hofmeister, L. und Stage, J. *Let your users do the testing: a comparison of three remote asynchronous usability testing methods*. 27th international conference on Human factors in computing systems, ACM, Boston, MA, USA, 2009, S. 1619-1628.

[Budde et al., 1992] Budde, R., Kantz, K., Kuhlenkamp, K. und Züllighoven, H. *Prototyping – An Approach to Evolutionary System Development.* Springer, Heidelberg, BRD, 1992.

[Burger et al., 2008] Burger, S., Burmester, M. und Selter, A. *Formatives Remote Usability Testing*. i-com, 1, 2008, S. 47-50.

[Buzan, 2006] Buzan, T. *Mind Map Book.* BBC Active, 2006.

[Cadiz et al., 2000] Cadiz, J.J., Gupta, A. und Grudin, J. *Using Web annotations for asynchronous collaboration around documents*. CSCW'00, Philadelphia, PA, USA, 2000, S. 309-318.

[Canning, 1956] Canning, R.G. *Electronic data processing for business and industry.* John Wiley and Sons, New York, 1956.

[Carey und Mason, 1983] Carey, T. und Mason, R.E.A. *Information system prototyping: techiques, tools, and methodologies*. INFOR - The Canadian Journal of Operational Research and Information Processing, 21 (3), 1983, S. 177-191.

[Carmel et al., 1993] Carmel, E., Whitaker, R.D. und George, J.F. *PD and joint application design: a transatlantic comparison*. Communications ofthe ACM, 36 (6), 1993, S. 40-48.

[Carrizo et al., 2008] Carrizo, D., Dieste, O. und Juristo , N. *Study of Elicitation Techniques Adequacy*. 11. Workshop on Requirements Engineering, Barcelona, ES, 2008.

[Carroll und Rosson, 1992] Carroll, J.M. und Rosson, M.B. *Getting around the task-artifact cycle: how to make claims and design by scenario*. ACM Transactions on Information Systems, 10, 1992, S. 181–212.

[Catlin et al., 1989] Catlin, T., Bush, P. und Yankelovich, N. *InterNote: extending a hypermedia framework to support annotative collaboration*. Hypertext 1989, ACM Press, 1989, S. 365-378.

[Chatzoglou und Macaulay, 1996] Chatzoglou, P.C. und Macaulay, L. *Requirements Capture and Analysis: A Survey of Current Practice*. Requirements Engineering, 1 (2), 1996, S. 75-87.

[Checkland, 1981] Checkland, P. *Systems Thinking, Systems Practice.* John Wiley & Sons, Chichester, USA, 1981.

[Chin-Yeh und Gwo-Dong, 2004] Chin-Yeh, W. und Gwo-Dong, C. Extending e-books with annotation, online support and assessment mechanisms to increase efficiency of learning. SIGCSE Bull., 36 (3), 2004, S. 132-136.

[Cioffi, 1986] Cioffi, G. *Relationships among comprehension strategies reported by college students*. Reading Research and Instruction, 25, 1986, S. 220-231.

[Clement und Van den Besselar, 1993] Clement, A. und Van den Besselar, P. *A Retrospective Look at PD Projects*. Communication of the ACM, 36 (4), 1993, S. 29-39.

[Cockburn, 2003] Cockburn, A. *Agile Software-Entwicklung.* Verlag Moderne Industrie, 2003.

[Converse und Presser, 1986] Converse, J.M. und Presser, S. *Survey Questions – Handcrafting the Standardardized Questionnaire.* Sage, Beverly Hills, USA, 1986.

[Cooper, 1999] Cooper, A. The immates are running the asylum: Why high-tech products drive us crazy and how to restore the sanitiy. SAMS, Indianapolis, IA, USA, 1999.

[Cosmos, 2000] Cosmos. *Cross-case analysis of transformed firms.* National Institute of Standards and Technology, U.S. Department of Commerce, Gaithersburg, USA, 2000, S. 111-123.

[Cox und Brna, 1995] Cox, R. und Brna, P. *Supporting the use of external representations in problem solving: the need for flexible learning environments*. Journal of Artificial Intelligence in Education, 6 (2/3), 1995, S. 239-302.

[Creighton et al., 2006] Creighton, O., Ott, M. und Bruegge, B. *Software Cinema - Video-based Requirements Engineering*. 14th IEEE International Requirements Engineering Conference (RE'06), Minneapolis/St. Paul, Minnesota, USA, 2006, S. 109-118.

[Damodaran, 1996] Damodaran, L. *User involvement in the systems design process – a practical guide for users.* . Behaviour & Information Technology, 15, 1996, S. 363–377.

[Danielson et al., 2006] Danielson, K., Naghsh, A.M. und Dearden, A. Distributed Participatory Design. Extended Abstract of the workshop for Distributed Participatory Design. NordiCHI 2006, 2006.

[Danielson et al., 2008] Danielson, K., Naghsh, A.M., Gumm, D. und Warr, A. *Distributed participatory design - Extended abstracts of the workshop session*. Human Factors in Computing Systems (CHI), Florence, ITA, 2008.

[Davis et al., 1997] Davis, A., Overmyer, S., Jordan, K., Caruso, J., Dandashi, F., Dinh, A., Kincaid, G., Ledeboer, G., Reynolds, P., Sitaram, P., Ta, A. und Theofanos, M. *Identifying and Measuring Quality in a Software Requirements Specification.* In: *Software Requirements Engineering,* Thayer, R.H. und Dorfman, M. (Hrsg). IEEE Computer Society Press, Washington, USA, 1997, S. 164-175.

[Davis et al., 2006] Davis, A., Dieste, O., Hickey, A., Juristo, N. und Moreno, A.M. *Effectiveness of Requirements Elicitation Techniques: Empirical Results Derived from a Systematic Review*. IEEE International Conference on Requirements Engineering, 2006, S. 179-188.

[Davis, 1990] Davis, A.M. Software Requirements - Analysis and Specification. Prentice-Hall, 1990.

[Davis und Huttenlocher, 1995] Davis, J.R. und Huttenlocher, D.P. *Shared Annotation for Cooperative Learning*. CSCL '95, Lawrence Erlbaum Associates, Bloomington, Indiana, USA, 1995, S. 84-88.

[Davis et al., 1999] Davis, R.C., Landay, J.A., Chen, V., Huang, J., Lee, R.B., Li, F.C., Lin, J., Morrey, C.B., Schleimer, B., Price, M.N. und Schilit, B.N. *NotePals: lightweight note sharing by the group, for the group*. SIGCHI conference on Human factors in computing systems: the CHI is the limit, ACM, Pittsburgh, Pennsylvania, USA, 1999, S. 338-345.

[Decker et al., 2007] Decker, B., Ras, E., Rech, J., Jaubert, P. und Rieth, M. *Wiki-based stakeholder participation in requirements engineering*. IEEE Software, 24 (2), 2007, S. 28–35.

[DeMarco, 1979] DeMarco, T. *Structured Analysis and System Specification.* Prentice Hall, 1979.

[Diaper, 2001] Diaper, D. *Task analysis for knowledge descriptions (TADK): a requiem for a method*. Behaviour & Information Technology, 20, 2001, S. 199–212.

[Dieste et al., 2008] Dieste, O., Lopez, M. und Ramos, F. *Updating a Systematic Review about Selection of Software Requirements Elicitation Techniques*. 11th Workshop on Requirements Engineering, 2008.

[Dieste und Juristo, 2010] Dieste, O. und Juristo, N. *Systematic Review and Aggregation of Empirical Studies on Elicitation Techniques*. IEEE Transactions on Software Engineering, 99, 2010.

[Dillmann, 1978] Dillmann, D.A. *Mail and Telephone Surveys - The Total Design Method.* Wiley, New York, USA, 1978.

[DIN, 2004] DIN, D.I.f.N. *Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten.* Deutsches Institut für Normung e.V., Berlin, BRD, 2004.

[Dmitriev et al., 2006] Dmitriev, P.A., Eiron, N., Fontoura, M. und Shekita, E. *Using annotations in enterprise search*. Proceedings of the 15th international conference on World Wide Web, ACM, Edinburgh, Scotland, 2006.

[Dodd und Carr, 1994] Dodd, J.L. und Carr, H.H. *Systems development led by end-users*. Joumal of Systems Management, 45 (8), 1994, S. 34-30.

[Döring, 2003] Döring, N. Sozialpsychologie des Internet. Die Bedeutung des Internet für Kommunikationsprozesse, Identitäten, soziale Beziehungen und Gruppen. 2. Auflage, Hogrefe-Verlag, 2003.

[Dörner et al., 2008] Dörner, C., Heß, J. und Pipek, V. *Fostering user-developer collaboration with infrastructure probes*. International workshop on Cooperative and human aspects of software engineering at the International Conference on Software Engineering, ACM, Leipzig, BRD, 2008.

[Dray und Siegel, 2004] Dray, S. und Siegel, D. *Remote possibilities?: international usability testing at a distance*. interactions, 11 (2), 2004, S. 10-17.

[Dray und Karat, 1994] Dray, S.M. und Karat, C. *Human factors cost justification for an internal development project.* In: *Costjustifying Usability,* Bias, R.G. und Mayhew, D.J. (Hrsg). Academic Press, San Diego, CA, USA, 1994.

[Duden, 2007] Duden. *Duden - Das Herkunftswörterbuch: Etymologie der deutschen Sprache.* Bibliographisches Institut, Mannheim, BRD, 2007.

[Ehn und Sandberg, 1979] Ehn, P. und Sandberg, Å. Management Control and Wage Earner Power (Foretagsstyrning och Lontagarmakt). Prisma, Falkoping, 1979.

[Ehn, 1989] Ehn, P. *Work Oriented Design of Computer Artifacts.* Lawrence Erlbaum, Hillsdale, 1989.

[Ehn, 1993] Ehn, P. *Scandinavian Design: On Participation and Skill.* In: *Participatory Design: Principles and Practices,* Schuler, D. und Namioka, A. (Hrsg). Lawrence Erlbaum, 1993, S. 41–78.

[Eisenhardt, 1989] Eisenhardt, K.M. *Building Theories from Case Study Research*. Acadamy of Management Review, 14, 1989, S. 532-550.

[El Emam et al., 1996] El Emam, K., Quintin, S. und Madhavji, N.H. *User Participation in the Requirements Engineering Process: An Empirical Study*. Requirements Engineering, 1 (1), 1996, S. 4-26.

[Erichson, 1995] Erichson, B. *Experimente.* In: *Handwörterbuch des Marketing,* Tietz, B., Köhler, R. und Zentes, J. (Hrsg). Schäffer-Poeschel, Stuttgart, BRD, 1995, S. 639-654.

[Fass und Schumacher, 1978] Fass, W. und Schumacher, G.M. *Effects of motivation, subject activity, and readability on the retention of prose materials*. Journal of Educational Psychology, 70 (5), 1978, S. 803-807.

[Fetterman, 1999] Fetterman, D.M. *Ethnography Step by Step.* SAGE Publications, London, GBR, 1999.

[Fischer et al., 2004] Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A.G. und Mehandjiev, N. *Meta-Design: A manifesto for end-user development*. Communication of the ACM, 47 (9), September, 2004, S. 33-37.

[Fish et al., 1988] Fish, R.S., Kraut, R.E. und Leland, M.D.P. *Quilt: a collaborative tool for cooprative writing*. Conference on Office information systems, ACM Press, 1988, S. 30-37.

[Fishbein und Ajzen, 1975] Fishbein, M. und Ajzen, I. *Belief, Attitude, Intention, and Behavior: An Introduction to Theory and Research.* Addison-Wesley, Boston, MA, USA, 1975.

[Fogli et al., 2004] Fogli, D., Fresta, G. und Mussio, P. *On Electronic Annotation and Its Implementation*. ACM AVI, May, 2004, S. 98-102.

[Foster und Franz, 1999] Foster, S.T. und Franz, C.R. *User involvement during information systems development: a comparison of analyst and user perceptions of system acceptance*. Journal of Engineering and Technology Management, 16 (3-4), 1999, S. 329-348

[Galletta und Lederer, 1989] Galletta, D.F. und Lederer, A.L. *Some Cautions on the Measurement of User Information Satisfaction**. Decision Sciences, 20 (3), 1989, S. 419-434.

[Garrison, 2003] Garrison, D.R. Cognitive presence for effective asynchronous online learning: The role of reflective inquiry, self-direction and metacognition. In: Elements of Quality Online Education: Practice and Direction, Bourne, J.M., J. C. (Hrsg). Sloan Center for Online Education, Needham, MA, USA, 2003, S. 47–58.

[Gärtner und Wagner, 1996] Gärtner, J. und Wagner, I. *Mapping Actors and Agendas: Political Frameworks of Systems Design and Participation*. Human-Computer Interaction, 11 (3), 1996, S. 187–214.

[Gärtner, 1998] Gärtner, J. *Participatory Design in Consulting*. Computer Supported Cooperative Work, 4 (3-4), 1998.

[Gaver, 1999] Gaver, B., Dunne, T., and Pacenti, E. . *Design: Cultural probes*. ACM Interactions, 6 (1), 1999, S. 21-29.

[Gehrau, 2002] Gehrau, V. *Die Beobachtung in der Kommunikationswissenschaft.* Utb, Stuttgart, BRD, 2002.

[Geißler et al., 2004] Geißler, S., Hampel, T. und Keil-Slawik, R. Vom virtuellen Wissensraum zur Lernumgebung – Kooperatives Lernen als integrativer Ansatz für eine mediengestützte Bildung. i-com, 2, 2004.

[Gemmell et al., 2002] Gemmell, J., Bell, G., Lueder, R., Drucker, S. und Wong, C. *MyLifeBits: fulfilling the Memex vision*. Proceedings of the tenth ACM international conference on Multimedia, ACM, Juan-les-Pins, France, 2002.

[Gibson, 1977] Gibson, H.L. *Determining user involvement*. Journal of Systems Management, 1977, S. 20-22.

[Gill, 1962] Gill, A. *Introduction to the Theory of Finite-state Machines.* McGraw-Hill, New York, USA, 1962.

[Glinz, 2007] Glinz, M. *On Non-Functional Requirements*. 15th IEEE International Requirements Engineering Conference (RE 2007), IEEE, New Delhi, Indien, 2007.

[Glukhova et al., 2009] Glukhova, A., Hocken, C. und Klamma, R. *Community Driven Elicitation of Requirements with Entertaining Social Software*. Workshop Open Design Spaces supporting User Innovation (ODS '09), Second International Symposium on End User Development, Siegen, BRD, 2009.

[Goguen und Linde, 1993] Goguen, J.A. und Linde, C. *Techniques for Requirements Elicitation*. IEEE International Symposium on Requirements Engineering, IEEE, San Diego, CA , USA 1993, S. 152-164.

[Goldberg, 2009] Goldberg, A. *Bug writing Guidelines.* https://bugs.eclipse.org/bugs/bugwritinghelp.html, abgerufen am 10.03.2009, 2009.

[Gould und Lewis, 1985] Gould, J.D. und Lewis, C. *Designing for usability: Key principles and what designers think*. Communications of the ACM, 28 (3), 1985, S. 300-311.

[Gould und Salaun, 1987] Gould, J.D. und Salaun, J. *Behavoioral Experiments on Handmarkings*. CHI 1987, ACM Press, 1987, S. 175-181.

[Gould et al., 1997] Gould, J.D., Boies, S.J. und Ukelson, J. *How to Design Usable Systems.* In: *Handbook of Human-Computer Interaction,* Helander, Landauer und Prabhu (Hrsg). Elsevier B.V., 1997.

[Green und Petre, 1996] Green, T.R.G. und Petre, M. *Usability analysis of visual programming environments: a 'cognitive dimensions' framework*. Journal of Visual languages and Computing, 7, 1996, S. 131-174.

[Greenbaum und Kyng, 1991] Greenbaum, J. und Kyng, M. *Design at Work: Cooperative Design of Computer Systems.* Erlbaum Associates, Hillsdale, New York, USA, 1991.

[Greenbaum, 1995] Greenbaum, J. Windows on the Workplace: Computers, Jobs, and the Organization of Office Work in the Late Twentieth Century. Mondthly Review Press, New York, USA, 1995.

[Greiffenberg, 2003a] Greiffenberg, S. *Methodenentwicklung in Wirtschaft und Verwaltung.* TU Dresden, Dresden, Germany 2003a.

[Greiffenberg, 2003b] Greiffenberg, S. *Methoden als Theorien der Wirtschaftsinformatik*. 6. Internationale Tagung der Wirtschaftsinformatik 2003, Physica-Verlag, Dresden, Germany, 2003b, S. 947-967.

[Grønbæk et al., 1997] Grønbæk, K., Kyng, M. und Mogensen, P. *Toward a Cooperative Experimental System Development Approach.* In: *Computers and Design in Context,* Kyng, M. und Mathiassen, L. (Hrsg). MIT Press, Cambridge, MA, USA, 1997, S. 201–238.

[Gross et al., 2007] Gross, T., Koch, M. und Herczeg, M. *Computer-Supported Cooperative Work.* Oldenbourg, München, BRD, 2007.

[Grudin, 1991a] Grudin, J. *Interactive Systems: Bridging the Gaps Between Developers and Users*. IEEE Computer, 24 (4), 1991a, S. 59-69.

[Grudin, 1991b] Grudin, J. Systematic sources of suboptimal interface designin large product development organization. Human-Computer Interaction, 6 (2), 1991b, S. 147-196.

[Grudin, 1996] Grudin, J. *The organizational contexts of development and use*. ACM Computing Surveys (CSUR), 28 (1), 1996, S. 169-171.

[Gulliksen et al., 2003] Gulliksen, J., Göransson, B., Boivie, I., Blomkvist, J.P. und Cajander, Â. *Key Principles for User-Centred Systems Design*. Behaviour & Information Technology, 22 (6), 2003, S. 397-409.

[Haake et al., 2004] Haake, J., Schwabe, G. und Wessner, M. *CSCL-Kompendium: Lehr- und Handbuch zum computerunterstützten kooperativen Lernen.* Oldenbourg, München, BRD, 2004.

[Hackos und Redish, 1998] Hackos, J.T. und Redish, J.C. *User and task analysis for interface design.* Wiley, New York, USA, 1998.

[Hammann und Erichson, 2006] Hammann, P. und Erichson, B. *Marktforschung.* 6. Auflage, UTB, Stuttgart, BRD, 2006.

[Harper und Simpson, 1998] Harper, R.H.R. und Simpson, J. *Inside The Imf: An Ethnography Of Documents, Technology, And Organizational Action.* Academic Press, London, GBR, 1998.

[Harper, 2000] Harper, R.H.R. *The Organisation in Ethnography: A Discussion of Ethnographic Fieldwork Programs in CSCW*. Computer Supported Cooperative Work, 9, 2000, S. 239–264.

[Hartson et al., 1996] Hartson, H.R., Castillo, J.C., Kelso, J. und Neale, W.C. *Remote evaluation: the network as an extension of the usability laboratory*. Proceedings of the SIGCHI conference on Human factors in computing systems: common ground, ACM, Vancouver, British Columbia, CAN, 1996, S. 228-235.

[Hartson und Castillo, 1998] Hartson, H.R. und Castillo, J.C. *Remote evaluation for post-deployment usability improvement*. Working Conference on Advanced Visual Interface (AVI'98). L'Aquila, IT, 1998.

[Hasenkamp et al., 1997] Hasenkamp, U., Kirn, S. und Syring, M. *CSCW. Computer Supported Cooperative Work* Addison Wesley, München, BRD, 1997.

[Hawk und Dos Santos, 1991] Hawk, S.R. und Dos Santos, B.L. *Successful system development: the effect of situational factors on alternative user roles*. IEEE Transactions on Engineering Management, 38 (4), 1991, S. 316-327.

[He, 1995] He, J. Knowledge Impacts of User Participation: A Cognitive Perspective. 1995.

[He und King, 2008] He, J. und King, W. *The Role of User Participation in Information Systems Development: Implications from a Meta-Analysis*. Journal of Management Information Systems, 25 (1), 2008, S. 301-331.

[Hedberg, 1975] Hedberg, B. *Computer systems to support industrial democracy.* In: *Human Choice and Computers* Mumford, E. und Sackman, H. (Hrsg). North-Holland, Amsterdam, NL, 1975.

[Hegarty und Steinhoff, 1997] Hegarty, M. und Steinhoff, K. *Individual differences in use of diagrams as external memory in mechanical reasoning*. Learning & Individual Differences, 9 (1), 1997, S. 19-43.

[Hegarty und Kozhevnikov, 1999] Hegarty, M. und Kozhevnikov, M. *Spatial abilities, working memory, and mechanical reasonig.* In: *Visual and spatial reasoning in design,* Gero, J.S. und Tversky, B. (Hrsg). University of Sydney, Sydney, AUS, 1999.

[Heinbokel et al., 1996] Heinbokel, T., Sonnentag, S., Frese, M., Stolte, W. und Brodbeck, F.C. *Don't underestimate the problems of user centredness in software development projects - there are many!* Behaviour & Information Technology, 15 (4), 1996, S. 226-236.

[Hekkert und Van Dijk, 2001] Hekkert, P. und Van Dijk, M. *Designing from context.* In: *Designing in context,* Lloyd, P. und Christiaans, H. (Hrsg). Delft University Press Science, Delft, NL, 2001, S. 383-394.

[Hesse et al., 1992] Hesse, W., Merbeth, G. und Frölich, R. *Software-Entwicklung - Vorgehensmodelle, Projektführung, Produktverwaltung.* Oldenbourg München, BRD, 1992.

[Hew und Cheung, 2003] Hew, K.F. und Cheung, W.S. Evaluating the participation and quality of thinking of pre-service teachers in an asynchronous online discussion environment: Part 1. International Journal of Instructional Media, 30 (3), 2003, S. 247-262.

[Hickey und Davis, 2003] Hickey, A.M. und Davis, A.M. *Elicitation Technique Selection: How Do Experts Do It?* 11th IEEE International Requirements Engineering Conference 2003, S. 169 - 178.

[Hilbert und Redmiles, 1999] Hilbert, D.M. und Redmiles, D.F. *Separating the Wheat from the Chaff in Internet-Mediated User Feedback Expectation-Driven Event Monitoring*. SIGGROUP Buletin, 20 (1), April, 1999, S. 35.

[Hirschheim, 1989] Hirschheim, R. *User participation in practice: Experiences with participative systems design.* In: *Participation in Systems Development,* Knight, K. (Hrsg). GP Publishing, Columbia, Maryland, USA, 1989, S. 194-212.

[Holtzblatt und Jones, 1993] Holtzblatt, K. und Jones, S. *Contextual Inquiry: A Participatory Technique for System Design.* In: *Participatory Design: Principles and Practices,* Schuler, D. und Namioka, A. (Hrsg). Lawrence Erlbaum, 1993, S. 177–210.

[Holtzblatt und Beyer, 1995] Holtzblatt, K. und Beyer, H.R. *Requirements Gathering: The Human Factor*. Communication of the ACM, 38 (5), May, 1995, S. 30-32.

[Holtzblatt und Beyer, 1999] Holtzblatt, K. und Beyer, H.R. *Contextual design: using customer work models to drive systems design*. CHI '99 extended abstracts on Human factors in computing systems, ACM, Pittsburgh, Pennsylvania, USA, 1999.

[Hooimeijer und Weimer, 2007] Hooimeijer, P. und Weimer, W. *Modeling bug report quality*. 22. IEEE/ACM International Conference on Automated Software Engineering (ASE) 2007, 2007, S. 34-43.

[Hughes et al., 1994] Hughes, J., King, V., Rodden, T. und Andersen, H. *Moving out from the control room: ethnography in system design*. CSCW, ACM, 1994, S. 429 - 439.

[Hüttner, 2002] Hüttner, M. *Grundzüge der Marktforschung.* 7. Auflage, Oldenbourg, Wiesbaden, BRD, 2002.

[Iivari und Iivari, 2006] Iivari, J. und Iivari, N. *Varieties of User-Centeredness*. 39th IEEE Hawaii International Conference on System Sciences, IEEE, Hawaii, USA, 2006.

[Iivari, 2004] Iivari, N. Enculturation of User Involvement in Software Development Organizations - An Interpretive Case Study in the Product Development Context. NordiCHI '04, Tampere, FIN, 2004.

[ISO 13407, 1999] ISO 13407. Human-centered design processes for interactive systems. 1999.

[Ives und Olson, 1984] Ives, B. und Olson, M. *User involvement and MIS success: A review of research*. Management Science, 30 (5), 1984, S. 586-603.

[Jahoda et al., 1933] Jahoda, M., Lazarsfeld, P.F. und Zeisel, H. *Die Arbeitslosen von Marienthal. Ein soziographischer Versuch.* Verlag S. Hirzel., Leipzig, BRD, 1933.

[Jahoda et al., 1975] Jahoda, M., Lazarsfeld, P.F. und Zeisel, H. *Die Arbeitslosen vom Marienthal – Ein soziographischer Versuch.* 1. Auflage, Suhrkamp, Allensbach, BRD, 1975.

[Kahan und Koivunen, 2001] Kahan, J. und Koivunen, M.-R. *Annotea: an open RDF infrastructure for shared Web annotations*. 10th International Conference on World Wide Web, 2001, S. 623-632.

[Kappelman und McLean, 1992] Kappelman, L.A. und McLean, E.R. *Promoting Information System Success: The Respective Roles of User Participation and User Involvement*. Journal of Information Technology Management Informatics, 3 (1), 1992, S. 1-12.

[Kappelman, 1995] Kappelman, L.A. *Measuring User Involvement: A Diffusion of Innovation Perspective*. DATABASE Advances, 26 (2&3), 1995, S. 65-86.

[Karat, 1997] Karat, J. *Evolving the Scope of User-Centered Design*. Communication of the ACM, 40 (7), 1997, S. 33-38.

[Kawalek und Wood-Harper, 2002] Kawalek, P. und Wood-Harper, T. *The Finding of Thorns: User Participation in Enterprise System Implementation*. The DATA BASE for Advances in Information Systems, 33 (1), 2002, S. 13-22.

[Kaya, 2007] Kaya, M. *Verfahren der Datenerhebung.* In: *Methodik der empirischen Forschung,* Albers, S., Klapper, D., Konradt, U., Walter, A. und Wolf, J. (Hrsg). Gabler, Wiesbaden, BRD, 2007, S. 49-64.

[Keen, 1981] Keen, P.G.W. *Information systems and organizational change*. Communications of ACM, 14 (1), 1981, S. 24-33.

[Keil-Slawik und Selke, 1998] Keil-Slawik, R. und Selke, H. *Mythen und Alltagspraxis von Technik und Lernen*. Informatik-Forum, 12 (1), 1998, S. 9-17.

[Keil und Carmel, 1995] Keil, M. und Carmel, E. *Customer-Developer Links in Software Development*. Communications of the ACM, 38 (5), 1995, S. 43-51.

[Kensing, 1983] Kensing, F. The Trade Unions Influence on Technological Change. In: Systems Design For, With and By the Users, Briefs, U. (Hrsg). North Holland, 1983.

[Kensing und Blomberg, 1998] Kensing, F. und Blomberg, J. *Participatory Design: Issues and Concerns*. Comput. Supported Coop. Work, 7 (3-4), 1998, S. 167-185.

[Kensing et al., 1998] Kensing, F., Simonsen, J. und Bodker, S. *MUST – A Method for Participatory Design*. Human-Computer Interaction, 13 (2), 1998, S. 167-198.

[Kensing und Blomberg, 2003] Kensing, F. und Blomberg, J. *Participatory Design: Issues and Concerns.* In: *Methods and Practices in Participatory Design,* Kensing, F. (Hrsg). ITU Press Copenhagen, Copenhagen, DK, 2003, S. 365-387.

[Kieback et al., 1992] Kieback, A., Lichter, H., Schneider-Hufschmidt, M. und Züllighoven, H. *Prototyping in industriellen Software-Produkten*. Informatik Spektrum, 12, 1992, S. 65-77.

[Kienle, 2003] Kienle, A. Integration von Wissensmanagement und kollaborativem Lernen durch technisch unterstützte Kommunikationsprozesse. Universität Dortmund, Dortmund, BRD 2003.

[Kitchenham und Pickard, 1995] Kitchenham, B.A. und Pickard, L. *Case Studies for Method and Tool Evaluation*. IEEE Software, 12 (4), 1995, S. 55-62.

[Kitchenham, 1996] Kitchenham, B.A. Evaluating software engineering methods and tool - part 2: selecting an appropriate evaluation method - technical criteria. SIGSOFT Softw. Eng. Notes, 21 (2), 1996, S. 11-15.

[Kittur et al., 2008] Kittur, A., Chi, E.H. und Suh, B. *Crowdsourcing User Studies With Mechanical Turk*. CHI 2008, ACM, Florence, ITA, 2008.

[Knittle et al., 1986] Knittle, D.L., Ruth, S. und Patton Gardner, E. *Establishing user-centered criteria for information systems: A software ergonomics perspective*. Information & Management, 11, 1986, S. 163-172.

[Krallmann und Frank, 2002] Krallmann, H. und Frank, H. *Systemanalyse im Unternehmen.* 4. Auflage, Oldenbourg, München, BRD, 2002.

[Kruchten, 2000] Kruchten, P. *The Rational Unified Process: An Introduction.* Addison-Wesley Longman, Boston, MA, USA, 2000.

[Kujala, 2003] Kujala, S. *User Involvement: A Review of Benefits and Challenges*. Behaviour & Information Technology, 22 (1), 2003, S. 1-16.

[Kujala et al., 2005] Kujala, S., Kauppinen, M., Lehtola, L. und Kojo, T. *The Role of User Involvement in Requirements Quality and Project Success*. 13th IEEE International Conference on Requirements Engineering (RE '05), 2005.

[Kujala, 2008] Kujala, S. *Effective user involvement in product development by improving the analysis of user needs*. Behaviour & Information Technology, 27 (6), 2008, S. 457-473.

[Kyng und Mathiassen, 1982] Kyng, M. und Mathiassen, L. *Systems Development and Trade Union Activities, in.* In: *Information Society, for richer, for poorer,* Bjørn-Andersen, N. (Hrsg). North Holland, Amsterdam, NL, 1982.

[Laden und Gildersleeve, 1963] Laden, H.N. und Gildersleeve, T.R. *System design for computer applications.* John Wiley and Sons, New York, USA, 1963.

[Lamnek, 2005] Lamnek, S. *Qualitative Sozialforschung - Lehrbuch.* 4. Auflage, Beltz Psychologie, Weinheim, BRD, 2005.

[Lapadat, 2002] Lapadat, J.C. *Written Interaction: A Key Component in Online Learning*. J. Computer-Mediated Communication, 7 (4), 2002.

[Larkin und Simon, 1987] Larkin, J.H. und Simon, H.A. *Why a diagram is (sometimes) worth ten thousand words* Cognitive Science, 11, 1987, S. 65-99.

[Larkin, 1989] Larkin, J.H. *Display-based problem solving.* In: *Complex information processing: the impact of Herbert A. Simon,* Klahr, D. und Kotovsky, K. (Hrsg). Lawrence Erlbaum, Hillsdale, New Jersey, USA, 1989, S. 319-341.

[Lauenroth und Riechert, 2009] Lauenroth, K. und Riechert, T. Der SoftWiki-Ansatz für verteiltes Requirements Engineering mit großen Stakeholdergruppen. In: Agiles Requirements Engineering für Softwareprojekte mit einer großen Anzahl verteilter Stakeholder, Auer, S., Lauenroth, K., Lohmann, S. und Riechert, T. (Hrsg). Leipziger Beiträge zur Informatik, Leipzig, BRD, 2009.

[Lettl, 2004] Lettl, C. Die Rolle von Anwendern bei hochgradigen Innovationen. DUV, Wiesbaden, BRD, 2004.

[Lewis, 2008] Lewis, S. *Using online communities to drive commercial product development*. CHI '08 extended abstracts on Human factors in computing systems, ACM, Florence, ITA, 2008, S. 2039-2044.

[Liao et al., 2008] Liao, C., Guimbretière, F., Hinckley, K. und Hollan, J. *Papiercraft: A gesture-based command system for interactive paper*. ACM Transactions on Computer-Human Interaction (TOCHI), 14 (4), 2008, S. 1-27.

[Lin et al., 2002] Lin, J., Thomsen, M. und Landay, J. *A Visual Language for Sketching Large and Complex Interactive Designs*. CHI 2002, 2002.

[Lloyd et al., 2002] Lloyd, W.J., Rosson, M.B. und Arthur, J.D. *Effectiveness of Elicitation Techniques in Distributed Requirements Engineering*. Joint International Conference on Requirements Engineering (RE'02), IEEE, 2002.

[Lohmann und Rashid, 2008] Lohmann, S. und Rashid, A. *Fostering Remote User Participation and Integration of User Feedback in Software DevelopmentL*. International Workshop on the Interplay between Usability Evaluation and Software Development (I-USED 2008), Pisa, IT, 2008.

[Lohmann und Ziegler, 2008] Lohmann, S. und Ziegler, J. *Webbasierte Erfassung und Analyse von Nutzeranforderungen*. Mensch und Computer 2008, Oldenbourg, Lübeck, Germany, 2008.

[Lohmann et al., 2008] Lohmann, S., Ziegler, J. und Heim, P. *Involving End Users in Distributed Requirements Engineering*. Second Conference on Human-Centered Software Engineering, Pisa, IT, 2008.

[Lucas, 1974] Lucas, H.C. *Systems quality, user reactions, and the use of information systems*. Management Informatics, 3 (4), 1974, S. 207-212.

[Maalej et al., 2009] Maalej, W., Happel, H.-J. und Rashid, A. *When Users Become Collaborators: Towards Continuous and Context-Aware User Input*. ACM OOPSLA OnWard! 2009, Orlando, Florida, USA, 2009.

[Macaulay, 1993] Macaulay, L. *Requirements Capture as a Cooperative Activity*. IEEE International Symposium on Requirements Engineering, San Diego, CA, USA, 1993, S. 174-181.

[Malinowski, 1922] Malinowski, B. *Argonauts of the Western Pacific.* Dutton & Co. Inc., New York, USA, 1922.

[Mao et al., 2005] Mao, J.-Y., Vredenburg, K., Smith, P.W. und Carey, T. *The State of User-Centered Design Practice*. Communication of the ACM, 48 (3), 2005, S. 105-109.

[Margolis und Resnick, 1999] Margolis, M. und Resnick, D. *Third voice: Vox populi vox dei?* First Monday, , 4 (10), 1999.

[Markus, 1983] Markus, M.L. *Power, politics, and MIS implementation*. Communications of ACM, 26 (6), 1983, S. 43-44.

[Marshall, 1997] Marshall, C.C. *Annotation: from paper books to the digital library*. Second ACM International Conference on Digital Libraries (DL 97), ACM, Philadelphia, PA, USA, 1997.

[Marshall, 1998] Marshall, C.C. *Toward an ecology of hypertext annotation*. 9th ACM Conference on Hypermedia and Hypertext (Hypertext '98), Pittsburgh, PA, USA, 1998, S. 40-49.

[Marshall, 2000] Marshall, C.C. *The future of annotation in a digital (paper) world.* In: *Successes and failures of digital libraries,* Twidale, M., Harum, S. und Harmon, S. (Hrsg). University of Illinois Press, Urbana-Campai, Illinois, USA, 2000, S. 97-117.

[Marshall und Brush, 2002] Marshall, C.C. und Brush, A.J.B. *From personal to shared annotations*. CHI 2002, 2002, S. 812-813.

[Marshall und Brush, 2004] Marshall, C.C. und Brush, A.J.B. *Exploring the Relationship between Personal and Public Annotations*. Joint ACM/IEEE Conference on Digital Libraries 2004 (JCDL'04), ACM, Tucson, Arizona, USA, 2004, S. 349-357.

[Maurer, 1996] Maurer, H. *Hyperwave: The Next Generation Web Solution.* Addison-Wesley, 1996.

[Mayhew, 1999] Mayhew, D.J. The Usability Engineering Lifecycle: a practitioner's handbook for user interface design. Morgan Kaufmann, San Francisco, CA, USA, 1999.

[Mayring, 2002] Mayring, P. *Einführung in die qualitative Sozialforschung.* 5. Auflage, Beltz, Weinheim, BRD, 2002.

[Mayring, 2007] Mayring, P. *Qualitative Inhaltsanalyse.* 9. Auflage, Utb, Stuttgart, BRD, 2007.

[Mazur, 2004] Mazur, J. Conversation analysis for educational technologists: Theoretical and methodological issues for researching the structures, processes and meaning of on-line talk. In: Handbook of research for educational communications and technology Jonassen, D. (Hrsg). McMillan, New York, USA, 2004, S. 1073–1098.

[McKeen und Guimaraes, 1997] McKeen, J.D. und Guimaraes, T. *Succesful strategies for user participation in system development*. Journal of Management Information Systems, 14 (2), 1997, S. 133-150.

[Meier, 2001] Meier, C. *Ethnografie.* In: *CSCW Kompendium - Lehr- und Handbuch zum computerunterstützten kooperativen Arbeiten,* Schwabe, G., Streitz, N. und Unland, R. (Hrsg). Springer, Berlin, BRD, 2001, S. 46-53.

[Mejias, 2005] Mejias, U.A. Re-approaching Nearness: Online communication and its place in Praxis. First Monday, 10 (3), 2005.

[Menne, 1984] Menne, A. Einführung in die Methodologie – Elementare allgemeine wissenschaftliche Denkmethoden im Überblick. 2. Auflage, Wissenschaftliche Buchgesellschaft, Darmstadt, BRD, 1984.

[Mertens et al., 2001] Mertens, P., Back, A. und Becker, J. *Lexikon der Wirtschaftsinformatik [Taschenbuch]* 4. Auflage, Springer, Berlin, BRD, 2001.

[Meyer, 2003] Meyer, K.A. *Face-to-face versus threaded discussions: The role of time and higher-order thinking*. Journal of Asynchronous Learning Networks, 7 (3), 2003, S. 55-65.

[Mikkelsen und Aasly, 2001] Mikkelsen, G. und Aasly, J. *Concordance of information in parallel electronic and paper based patient records*. International Journal of Medical Informatics, 63 (3), 2001, S. 123-131.

[Miles und Huberman, 1994] Miles, M.B. und Huberman, A.M. *Qualitative Data Analysis.* Thousand Oaks, USA, 1994.

[Millen, 2000] Millen, D.R. *Rapid ethnography: time deepening strategies for HCI field research*. Designing interactive systems: processes, practices, methods, and techniques, ACM, New York City, New York, USA, 2000, S. 280 - 286.

[Mohan et al., 2006] Mohan, K., Xu, P., Cao, L. und Ramesh, B. Integrating Traceability and Software Conguration Management: Improving Change Management in Software Development. Decision Support Systems, 2006.

[Moore und Shipman, 2000] Moore, J.D. und Shipman, F. *A Comparison of Questionnaire-Based and GUI-Based Requirements Gathering*. 15th IEEE international conference on Automated software engineering, IEEE, 2000, S. 35-43.

[Moore und Shipman, 2001] Moore, J.M. und Shipman, F.M. *Requirements elicitation using visual and textual information*. Fifth IEEE International Symposium on Requirements Engineering Toronto, Ontario, CA, 2001.

[Moore, 2003] Moore, J.M. *Communicating Requirements Using End-User GUI Constructions with Argumentation*. Proceedings.of the 18th IEEE International Conference on Automated Software Engineering ASE'03, Montral, CA, 2003.

[Moreland et al., 1997] Moreland, J.L., Danserau, D.F. und Chmielewski, T.L. *Recall of descriptive information: the roles of presentation format, annotation strategy, and individual differences*. Contemporary educational psychology, 22, 1997, S. 521-533.

[Moret, 1982] Moret, B. *Decision trees and diagrams*. ACM Computer Surveys, 14 (4), 1982, S. 593-623.

[Muller, 1991] Muller, M. *PICTIVE: An exploration in participatory design*. CHI 1991, ACM, New Orleans, Louisiana, USA, 1991, S. 225-231.

[Muller, 1993] Muller, M. PICTIVE: Democratizing the Dynamics of the Design Session. In: Participatory Design: Principles and Practices Schuler, D. und Namioka, A. (Hrsg). Lawrence Erlbaum, 1993, S. 211–238.

[Muller et al., 1995] Muller, M., Tudor, L.B., Wildman, D.M., White, E.A., Root, R.W., Dayton, T., Carr, R., Diekmann, B. und Dykstr-Erickson, E.A. *Bifocal Tolls for Scenarios and Representations in Participatory Activities with Users.* In: *Scenario-based Design for Human-computer Interaction,* Carroll, J. (Hrsg). Wiley, New York, NY, USA, 1995.

[Muller et al., 1997] Muller, M.J., Hallewell Haslwanter, J. und Dayton, T. *Participatory practices in the software lifecycle.* In: *Handbook of human-computer interaction,* Helander, M., Landauer, T.K. und Prabhu, P. (Hrsg). Elsevier, Amsterdam, NL, 1997, S. 255–297.

[Mumford, 1995] Mumford, E. *Effective Systems Design and Requirement Analysis: The Ethics Method.* Palgrave Macmillan, Houndmills, GBR, 1995.

[Naghsh und Dearden, 2004] Naghsh, A.M. und Dearden, A. *GABBEH - A Tool to Support Collaboration in Electronic Paper Prototyping: A Demostration*. ACM Conference on Computer Supported Cooperative Work, Chicago, USA, 2004.

[Naghsh et al., 2005] Naghsh, A.M., Dearden, A. und Ozcan, M.B. *Investigating Annotation in Electronic Paper-Prototypes*. International Workshop on Design, Specification and Verification of Interactive Systems, Newcastle upon Tyne, GBR, 2005.

[Nardi, 1997] Nardi, B.A. *The use of ethnographic methods in design and evaluation.* In: *Handbook of Human-Computer Interaction,* Helander, M.G., Landauer, T. und Prabhu, P. (Hrsg). Elsevier Science, Amsterdam, NL, 1997, S. 361-366.

[Neuwirth et al., 1990] Neuwirth, C., Kaufer, D., Chandhok, R. und Morris, J.H. *Issues in the Design of Computer Support for Co-Authoring and Commenting*. CSCW 1990, 1990, S. 183-195.

[Nielsen, 1993] Nielsen, J. *Usability Engineering.* Academic Press, San Diego, CA, USA, 1993.

[Nielsen, 1994a] Nielsen, J. *Usability Engineering*. Elsevier Academic Press, 1994a.

[Nielsen, 1994b] Nielsen, J. *Heuristic evaluation.* In: *Usability Inspection Methods,* Nielsen, J.M., R.L. (Hrsg). John Wiley & Sons, New York, NY, USA, 1994b.

[Nielsen und Mack, 1994] Nielsen, J. und Mack, R.L. *Usability inspection methods.* Wiley, New York, NY, USA, 1994.

[Noack und Schienmann, 1999] Noack, J. und Schienmann, B. *Objektorientierte Vorgehensmodelle im Vergleich*. Informatik-Spektrum, 22, 1999, S. 166–180.

[Noël und Robert, 2003] Noël, S. und Robert, J.-M. *Empirical study on collaborative writing: What do co-authors do, use, and like?* Computer Supported Cooperative Work: The Journal of Collaborative Computing, 13 (1), 2003, S. 63-89.

[Nokelainen et al., 2003] Nokelainen, P., Kurhila, J., Miettinen, M., Floréen, P. und Tirri, H. *Evaluating the Role of a Shared Document-based Annotation Tool in Learner-centered Collaborative Learning*. 3rd IEEE International Conference on Advanced Learning Technologies (ICALT 2003), IEEE, Athen, GRE, 2003, S. 200--203.

[Nokelainen et al., 2004] Nokelainen, P., Miettinen, M., Kurhila, J., Floréen, P. und Tirri, H. *A shared document-based annotation tool to support learner-centered collaborative learning.* Helsinki Institute for Information Technology HIIT, 2004.

[Norman, 1986] Norman, D.A. *Cognitive engineering.* In: *User-Centered Design: New Perspectives in Human-Computer Interaction,* D.A., N. und Draper, S.W. (Hrsg). Lawrence Erlbaum Associates, Hillsdale, New Jersey, USA, 1986.

[Norton und McFarlan, 1975] Norton, D. und McFarlan, F.W. *Product management.* In: *The Information Systems Handbook,* Nolan, R.L. und McFarlan, F.W. (Hrsg). Dow Jones-Irwin, Homewood, IL, USA, 1975, S. 517-528.

[Noyes et al., 1996] Noyes, J.M., Starr, A.F. und Frankish, C.R. *User Involvement in the Early Stages of the Development of an Aircraft Warning System*. Behaviour & Information Technology, 15 (2), 1996, S. 67-75.

[Nygaard und Bergo, 1975] Nygaard, K. und Bergo, O.T. *Trade unions: new users of research*. Personnel Review, 4, 1975, S. 2.

[Nygaard, 1979] Nygaard, K. *The Iron and Metal Project: Trade Union Participation.* Swedish Center for Working Life, Malmø, SWE, 1979.

[O'Hara und Sellen, 1997] O'Hara, K. und Sellen, A.J. *A Comparison of Reading Paper and On-Line Documents*. Conference on Human Factors in Computing Systems (CHI'97), 1997, S. 150-170.

[O'Neill et al., 1995] O'Neill, E., Johnson, P. und Coulooouris, G. *CUSTARD: A participatory approach to task analysis, software requirements synthesis and design*. Computers in Context: Joining Forces in Design, Aarhus, Denmark, 1995.

[Olson und Ives, 1981] Olson, M. und Ives, B. *User involvement in system design: An empirical test of alternative approaches*. Information & Management, 4 (4), 1981, S. 183-195.

[Ortlieb und Holz auf der Heide, 1993] Ortlieb, S. und Holz auf der Heide, B. Benutzer bei der Software-Entwicklung angemessen beteiligen - Erfahrungen und Ergebnisse mit verschiedenen Konzepten. In: Software-Ergonomie '93 Von der Benutzeroberfläche zur Arbeitsgestaltung - Berichte des German Chapter of the ACM, Rödiger, K.-H. (Hrsg). 1993, S. 249-261.

[Ovsiannikov et al., 1999] Ovsiannikov, I.A., Arbib, M.A. und Mcneill, T.H. *Annotation technology*. Int. J. Human-Computer Studies, January, 1999, S. 329-362.

[Panne et al., 2003] Panne, G.v.d., Beers, C.v. und Kleinknecht, A. *Success and failure of innovation: A literature review*. International Journal of Innovation Management, 7 (3), 2003, S. 309-338.

[Payne, 1951] Payne, S. *The Art of Asking Questions.* Princeton University Press, Princeton, USA, 1951.

[Pekkola et al., 2006] Pekkola, P., Kaarilahti, N. und Pohjola, P. *Participatory Design and ISD Methodologies*. Ninth Participatory Design Conference 2006 (PDC '06), Trento, ITA, 2006.

[Peterson, 1977] Peterson, J. *Petri nets*. ACM Computer Surveys, 9 (3), 1977, S. 223-252.

[Petre und Green, 1993] Petre, M. und Green, T.R.G. *Learning to read graphics: some evidence that 'seeing' an informational display is an aquired skill*. Journal of Visual Languages and Computing, 4 (55-70), 1993

[Pfleeger, 1998] Pfleeger, S.L. *Software Engineering: Theory & Practice.* Pearson US Imports & PHIPEs, 1998.

[Plimmer und Apperley, 2003] Plimmer, B. und Apperley, M. *FreeForm: A tool for sketching form designs*. HCI 2003, 2003.

[PMI, 2003] PMI, P.M.I. *A Guide to the Project Management Body of Knowledge.* Newtown Square, Pennsylvania, USA, 2003.

[Prause et al., 2008] Prause, C.R., Scholten, M., Zimmermann, A., Reiners, R. und Eisenhauer, M. *Managing the Iterative Requirements Process in a Multi-national Project Using an Issue Tracker*. IEEE International Conference on Global Software Engineering 2008, IEEE Computer Society, 2008, S. 151-159.

[Rashid et al., 2006a] Rashid, A., Behm, A., Müller-Arnold, T. und Rathgeb, M. *Kollaborative Benutzerbeteiligung im Requirements Engineering*. Mensch und Computer 2006, 2006a.

[Rashid et al., 2006b] Rashid, A., Meder, D., Wiesenberger, J. und Behm, A. *Visual Requirement Specification In End-User Participation*. IEEE International Conferece on Requirements Engineering (RE), 2006b.

[Rashid, 2007] Rashid, A. OpenProposal: Grafisches Annotieren von Verbesserungsvorschlägen für Software. Mensch & Computer 2007, 2007.

[Rashid, 2009] Rashid, A. OpenProposal: Ein Werkzeug zum Annotieren von "Snapshots" zum Erstellen von Anwendervorschlägen. Softwaretechniktrends, 29 (1), 2009.

[Rashid und Dinther, 2009] Rashid, A. und Dinther, C. Annotieren von Bildschirmfotos zur Steigerung der Anwenderbeteiligung. Mensch und Computer 2009, Berlin, BRD, 2009.

[Raymond, 2001] Raymond, E.S. *The Cathedral & the Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary.* 2. Auflage, O'Reilly & Associates, Sebastopol, CA, USA, 2001.

[Raza et al., 2010] Raza, A., Capretz, L.F. und Ahmed, F. *Improvement of Open Source Software Usability: An Empirical Evaluation from Developers' Perspective*. Advances in Software Engineering, 2010 2010.

[Riedewald, 2003] Riedewald, S. *Annotieren von grafischen Darstellungen auf dem Bildschirm.* Albert-Ludwig-Universität Freiburg i. Br. 2003.

[Robertson und Robertson, 1998] Robertson, J. und Robertson, S. *Volere Requirements Specification Template.* Atlantic Systems Guild, 1998.

[Robertson und Robertson, 2005] Robertson, S. und Robertson, J. *Requirements-led project management.* Addison-Wesley, 2005.

[Robey und Farrow, 1982] Robey, D. und Farrow, D.L. *User involvement in information system development: a conflict model and empirical test*. Management Science, 28 (1), 1982, S. 73-85.

[Röscheisen et al., 1994] Röscheisen, M., Mogensen, C. und Winograd, T. Shared Web annotations as a platform for third-party valueadded information providers: Architecture protocols and usage examples Stanford University Stanford, USA, 1994.

[Röscheisen et al., 1995] Röscheisen, M., Mogensen, C. und Winograd, T. *Beyond browsing: shared comments, SOAPs, trails, and on-line communities*. Third International World-Wide Web conference on Technology, tools and applications, Elsevier, Darmstadt, BRD, 1995, S. 739-749.

[Ross, 1977] Ross, D.T. *Structured Analysis (SA). A language for communiciating ideas.* IEEE Transactions on Software Engineering, SE-3 (1), 1977, S. 16-34.

[Royce, 1970] Royce, W. Managing the development of large software systems. 1970, S. 1-9.

[Royce, 1986] Royce, W. *Managing the development of large software systems*. International Conference on Software Engineering, IEEE Computer Society Press, Washington DC, 1986, S. 328-338.

[Rubin, 1994] Rubin, J. Handbook of usability testing: how to plan, design, and conduct effective tests. 1. Auflage, John Wiley & Sons, New York, USA, 1994.

[Rubin et al., 2008] Rubin, J., Chisnell, D. und Spool, J. *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests.* 2. Auflage, John Wiley & Sons, New York, USA, 2008.

[Rupp, 2001] Rupp, C. Requirements Engineering und Management, professionelle Anforderungsanalyse für die Praxis Hanser Verlag, 2001.

[Sarodnick und Brau, 2006] Sarodnick, F. und Brau, H. *Methoden der Usability Evaluation.* Verlag Hans Huber, Göttingen, BRD, 2006.

[Schickler et al., 1996] Schickler, M., Mazer, M. und Brooks, C. *Pan-Browser Support for Annotations and Other Meta-Information on the WWW* Special Issue of Computer Networks and ISDN Systems, 28 (7-11), 1996, S. 1063-1074.

[Schilit et al., 1998] Schilit, B.N., Golovchinsky, G. und Price, M.N. *Beyond paper: supporting active reading with free form digital ink annotations*. CHI 98, ACM Press, 1998.

[Schnell et al., 2008] Schnell, R., Hill, P.B. und Esser, E. *Methoden der empirischen Sozialforschung.* 8. Auflage, Oldenbourg Verlag, München, BRD, 2008.

[Schoman und Ross, 1977] Schoman, K. und Ross, D.T. *Structured analysis for requirements definition*. IEEE Transactions on Software Engineering, SE-3 (1), 1977, S. 6-15.

[Schönthaler, 1989] Schönthaler, F. Rapid prototyping zur Unterst¨utzung des konzeptuellen Entwurfs von Informationssystemen. Universität Karlsruhe (TH), Karlsruhe 1989.

[Schulmeister, 2006] Schulmeister, R. *eLearning: Einsichten und Aussichten.* Oldenbourg, München, BRD, 2006.

[Schwabe et al., 2001] Schwabe, G., Streitz, N. und Unland, R. *CSCW-Kompendium: Lehr- und Handbuch zum computerunterstützten kooperativen Arbeiten.* 1. Auflage, Springer, Berlin, BRD, 2001.

[Sharples et al., 1993] Sharples, M., Goodlet, J.S., Beck, E.E., Wood, C.C., Easterbrook, S.M. und Plowman, L. *Research issues in the study of computer supported collaborative writing.* In: *Computer Supported Collaborative Writing,* Sharples, M. (Hrsg). Springer, Heidelberg, Berlin, BRD, 1993, S. 9-28.

[Shneiderman und Plaisant, 2005] Shneiderman und Plaisant. *Usability Testing.* Pearson Education, Maryland, USA, 2005.

[Shneiderman, 2002] Shneiderman, B. *Leonardo's Laptop: Human Needs and New Computing Technologies.* MIT Press, Cambridge, MA, USA, 2002.

[Shotsberger, 2000] Shotsberger, P.G. *The Human Touch: Synchronous Communication in Web-Based Learning*. Educational Technology, 2000, S. 53-55.

[Sommerville, 1996] Sommerville, I. *Software Engineering.* Addison-Wesley, Edinburgh, GBR, 1996.

[Spolsky, 2004] Spolsky, J. *Joel on Software.* Acadamic Press, 2004.

[Sridhar et al., 2009] Sridhar, V., Nath, D. und Malik, A. *Analysis of User Involvement and Participation on the Quality of IS Planning Projects: An Exploratory Study*. Journal of Organizational and End User Computing, 21 (3), 2009, S. 80-98.

[Stahlknecht und Hasenkamp, 2005] Stahlknecht, P. und Hasenkamp, U. *Einführung in die Wirtschaftsinformatik.* 11. Auflage, Springer, 2005.

[Stake, 1995] Stake, R.E. *The Art of Case Study Research.* 1. Auflage, SAGE Publications, Thousand Oaks, USA, 1995.

[Stake, 2006] Stake, R.E. *Multiple Case Study Analysis.* Guilford, New York, USA, 2006.

[Steen et al., 2007] Steen, M., Kuijt-Evers, L. und Klok, J. *Early user involvement in research and design projects – A review of methods and practices*. 23rd EGOS Colloquium (European Group for Organizational Studies), Wien, AU, 2007.

[Stevens und Draxler, 2006] Stevens, G. und Draxler, S. *Partizipation im Nutzungskontext*. Mensch & Computer 2006, Oldenbourg Verlag, Gelsenkirchen, BRD, 2006, S. 83- 92.

[Stewart und Williams, 2005] Stewart, J. und Williams, R. The wrong trousers? Beyond the design fallacy: Social learning and the user. In: User involvement in innovation processes:Strategies and limitations from a socio-technical perspective, Rohracher, H. (Hrsg). Profil Verlag, Munchen, BRD, 2005, S. 39-71.

[Suchman, 1995] Suchman, L. *Making Work Visible*. Communication of the ACM, 38 (9), 1995, S. 56-64.

[Suthers und Xu, 2002] Suthers, D. und Xu, J. *Kukakuka: An Online Environment for Artefact-Centered Discourse* 11th International World-Wide Web Conference, ACM, 2002.

[Teichrow und Hershey, 1977] Teichrow, D. und Hershey, E.A. *PSL/PSA: a computer aided technique for structured documentation and analysis of information process systems*. IEEE Transactions on Software Engineering, SE-3 (1), 1977, S. 41-48.

[Thompson et al., 2004] Thompson, K.E., Rozanski, E.P. und Haake, A.R. *Here, there, anywhere: remote usability testing that works*. Proceedings of the 5th conference on Information technology education, ACM, Salt Lake City, UT, USA, 2004, S. 132-137.

[Tierney und Pearson, 1983] Tierney, R.J. und Pearson, P.D. *Toward a composing model of reading*. Language Arts, 60, 1983, S. 568–680.

[Tohidi et al., 2006] Tohidi, M., Buxton, W., Baecker, R. und Sellen, A. *Getting the Right Design and the Design Right: Testing Many is Better Than One*. CHI 2006, ACM Press, 2006.

[Trafton und Trickett, 2001] Trafton, J.G. und Trickett, S.B. *Note-Taking for self-explanation and problemsolving*. International Journal of Human Computer interaction, 16, 2001, S. 1-38.

[Tripp, 1998] Tripp, L.L. IEEE Guide for Developing System Requirements Specifications. IEEE, 1998.

[Tullis et al., 2002] Tullis, T., Fleischman, S., McNulty, M., Cianchette, C. und Bergel, M. *An Empirical Comparison of Lab and Remote Usability Testing of Web Sites*. Usability Professionals Association Conference (UPA), 2002.

[Van Maanen und Kolb, 1986] Van Maanen, J. und Kolb, D. The professional apprentice: observations on fieldwork roles in two organizational settings. In: Research in the sociology of organizations, Bacharach, S.B. und Mitchell, S.M. (Hrsg). JAI Press, 1986, S. 1-33.

[Virzi, 1992] Virzi, R.A. Refining the test phase of usability evaluation: how many subjects is enough? Human Factors, 34, 1992, S. 457-468.

[Vo, 2007] Vo, H.T.K. *Engineering Corporate Portals.* Universität Karlsruhe, Karlsruhe, BRD 2007.

[Vo und Elsner, 2007] Vo, H.T.K. und Elsner, H. *Management of Portal Evolution*. 8.Internationale Tagung Wirtschaftsinformatik, Universitätsverlag Karlsruhe, Karlsruhe, BRD, 2007, S. 337-354.

[Vredenburg et al., 2001] Vredenburg, K., Isensee, S. und Righi, C. *User-Centered Design: An Integrated Approach* Prentice Hall, 2001.

[Walldius et al., 2009] Walldius, Å., Sundblad, Y., Bengtsson, L., Sandblad, B. und Gulliksen, J. *User certification of workplace software: assessing both artefact and usage*. Behaviour & Information Technology, 28 (2), 2009, S. 101-120.

[Walter und Nagypal, 2008] Walter, A. und Nagypal, G. *Efficient Integration of Semantic Technologies for Professional Image Annotation and Search*. IADIS international conference e-Society 2008, Algarve, Portugal, 2008, S. 187-194.

[Weidenmann, 1994] Weidenmann, B. *Wissenserwerb mit Bildern.* Huber, Bern, SUI, 1994.

[Weltz und Ortmann, 1992] Weltz, F. und Ortmann, R. *Das Softwareprojekt - Projektmanagement in der Praxis.* Campus, Frankfurt, BRD, 1992.

[Weng und Gennari, 2004] Weng, C. und Gennari, J. *Asynchronous Collaborative Writing through Annotations, Notes*. CSCW 2004, ACM Press, 2004, S. 564-573.

[Whitehead, 2007] Whitehead, J. *Collaboration in Software Engineering: A Roadmap*. Future of Software Engineering (FOSE'07), 2007.

[Wilcox et al., 1997] Wilcox, L.D., Schilit, B.N. und Sawhney, N. *Dynomite: a dynamically organized ink and audio notebook*. CHI 97, ACM Press, Atlanta, GA, USA, 1997.

[Wilson et al., 1996] Wilson, A., Bekker, M., Johnson, H. und Johnson, P. *Costs and benefits of user involvement in design: Practitioners' views*. HCI'96, Springer, London, GBR, 1996.

[Wilson et al., 1997] Wilson, A., Bekker, M., Johnson, P. und Johnson, H. *Helping and Hindering User Involvement - A Tale of Everyday Design*. Conference on Human Factors in Computing Systems, 1997, S. 178-185.

[Wixon et al., 1994] Wixon, D., Jones, S., Tse, L. und Casady, G. *Inspections and design reviews: framework, history, and reflection.* In: *Usability inspection methods,* Nielsen, J. und Mack, R.L. (Hrsg). Wiley, New York, USA, 1994, S. 77–103.

[Wixon und Ramey, 1996] Wixon, D. und Ramey, J. *Field methods casebook for software design.* Wiley, New York, USA, 1996.

[Wixon et al., 2002] Wixon, D.R., Ramey, J., Holtzblatt, K., Beyer, H., Hackos, J., Rosenbaum, S., Page, C., Laakso, S.A. und Laakso, K. *Usability in practice: field methods evolution and revolution*. CHI' 2002, ACM Press, Minneapolis, Minnesotta, USA, 2002, S. 880–884.

[Wood und Silver, 1995] Wood, J. und Silver, D. *Joint Application Development.* 2. Auflage, Wiley, 1995.

[Yee, 2002] Yee, K.-P. *CritLink: Advanced Hyperlinks Enable Public Annotation on the Web* CSCW 2002, ACM, New Orleans, Louisiana, USA, 2002.

[Yin, 2003] Yin, R.K. *Case Study Research: Design and Methods.* 3. Auflage, SAGE Publications, Thousand Oaks, USA, 2003.

[Zeffane et al., 1998] Zeffane, R., Cheek, B. und Meredith, P. *Does user involvement during information systems development improve data quality?* . Human Systems Management, 17 (2), 1998, S. 115-121.

[Zheng et al., 2006] Zheng, Q., Booth, K. und McGrenere, J. *Co-Authoring with Structured Annotations*. CHI 2006, ACM Press, Montréal, Québec, CAN, 2006, S. 131-140.

Anhang

Anhang A: Versionen von OpenProposal

Die Entwicklung von OpenProposal wurde durch zahlreiche tiefgreifende Entscheidungen bezüglich der Gestaltung des Werkzeuges geprägt, so dass das Funktionsangebot und die Bedienoberfläche in jeder neuen Version verändert wurden. Dies ist darin begründet, dass die Komponenten und die Bedienoberfläche von OpenProposal bei jedem Benutzertest aufs Neue in Frage gestellt wurden, um OpenProposal so kontinuierlich zu verbessern. Außerdem gehörte es zur Vorgehensweise, mögliche alternative Lösungsansätze zu testen und die jeweils am besten geeignete beizubehalten.

Die erste Implementierung wurde im Jahr 2006 in Form einer Machbarkeitsstudie durchgeführt und hatte den Zweck, die grundlegende Idee von OpenProposal in einer ersten Vision abzubilden. Mit dieser Version wurde der Grundstein für die zukünftigen Versionen gesetzt: Erste Funktionalitäten zum Erstellen von Bildschirmfotos, zum Zeichnen grafischer Elemente (z.B. Linie, Kreis, Rechteck) und zum Platzieren von Text wurden zur Verfügung gestellt. Die Implementierung erfolgte damals in Java. Dabei wurde festgestellt, dass das Auslesen von Systemparameter aus dem Betriebssystem mit Java erschwert bzw. unmöglich ist, weswegen die Umsetzung auf C# portiert wurde. In Abbildung 10.3 ist die damalige Benutzeroberfläche abgebildet.

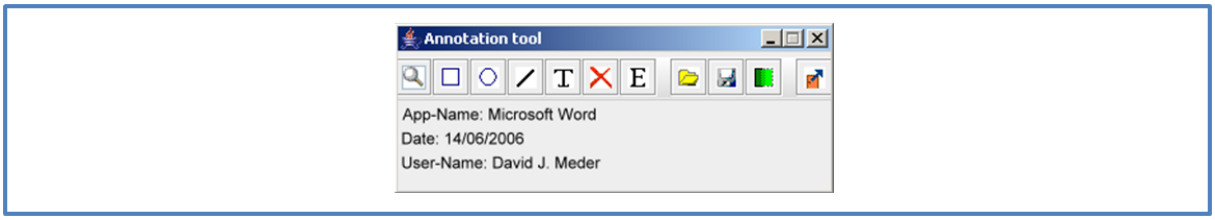

Abbildung 10.3: OpenProposal in Version 0.2

Version 1.0 erschien Anfang 2007. Die grafische Benutzeroberfläche wurde stark überarbeitet und es wurden neben grafischen Zeichenwerkzeugen auch die ersten Annotationswerkzeuge (Hinzufügen, Streichen, Verschieben, Kommentieren) implementiert. Während eines ersten Benutzertests im Februar 2007 mit 15 Testpersonen wurde das Prinzip der Annotation von Bildschirmfotos getestet und untersucht, ob Anwender es vorziehen frei zu zeichnen oder das Verwenden von Annotationswerkzeugen bevorzugen. Dabei zeigte sich, dass Annotationswerkzeuge als sinnvoller und einfacher zu bedienen bewertet wurden. Insgesamt reagierten die Testpersonen sehr positiv auf OpenProposal und brachten viele neue Ideen ein. Die damalige Oberfläche wird in Abbildung 10.4 dargestellt.

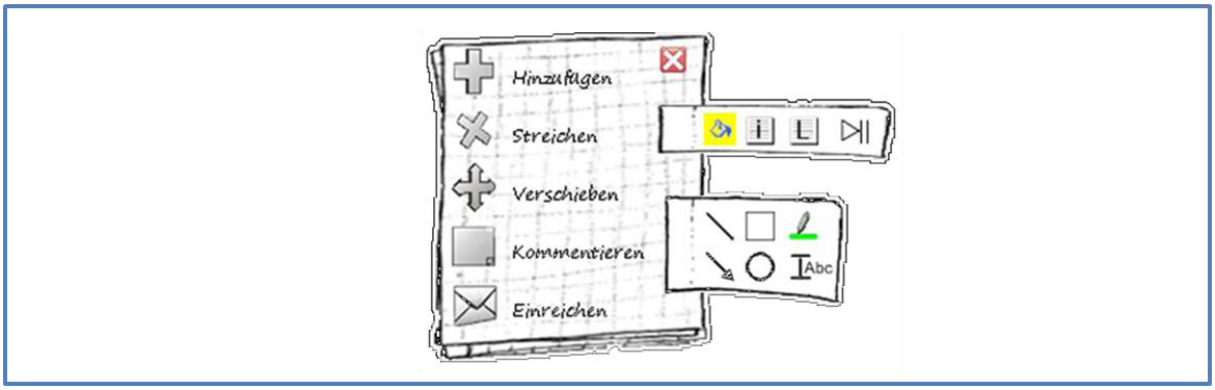

Abbildung 10.4: OpenProposal in Version 1.0

Im Anschluss an die ersten Benutzertests wurde Mitte 2007 Version 2.0 fertiggestellt. Abbildung 10.5 bildet die damalige Benutzeroberfläche ab, deren grafische Darstellung stark überarbeitet wurde.

Außerdem wurde eine Dreiteilung des Annotationsprozesses implementiert: So muss der Anwender zunächst die Kategorie des Vorschlags auswählen(1), anschließend können die Annotationen erstellt werden (2). Zum Schluss müssen ein Titel sowie eine kurze Beschreibung abgegeben werden (3). Weiterhin wurden eine Objektliste (4) und eine Undo-Funktionalität (5) implementiert.

Im Rahmen früherer Tests war festgestellt worden, dass die Anwender Schwierigkeiten hatten, wenn sie an einem Bildschirmfoto noch etwas nachträglich ändern wollten. So war das Problem aufgetreten, dass ein Programmfenster ein anderes Fenster verdeckte, welches wichtige Informationen enthielt. Daher wurde in dieser Version eine Funktionalität (6) implementiert, die es erlaubte, die Annotation zu pausieren und das störende Fenster zu verschieben.

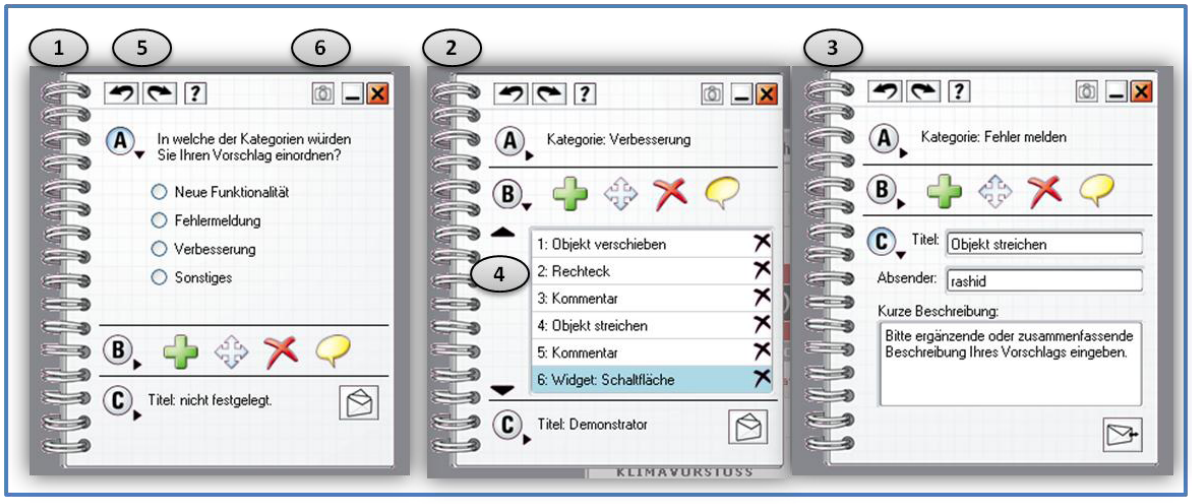

Abbildung 10.5: OpenProposal in Version 2.0

In Version 2.1 wurde im Herbst 2007 die Anbindung an Issue-Tracker und Email realisiert sowie kleine Anpassungen der grafischen Benutzeroberfläche durchgeführt. Abbildung 10.6 bildet die damalige Version ab. Diese Version wurde im Rahmen der Fallstudie TRUMPF eingesetzt.

Wird das Programm gestartet, ist der Abschnitt A (A) sichtbar, Abschnitte B und C zeigen nur die nötigsten Informationen. Im blauen Abschnitt unterhalb des Hauptfensters werden stets Hinweise und Hilfen zu den gerade möglichen Aktionen angezeigt (G). Der Benutzer kann nun die Art des Vorschlages wählen oder direkt in einen anderen Abschnitt springen. Klickt der Benutzer mit der Maus auf das Start Symbol wird ein neuer Screenshot erstellt und in den Annotationsmodus gewechselt. Das Start Symbol wird zum Pause Symbol (B). Auch ein Mausklick auf eines der vier Werkzeuge (1, 2, 3 und 4) erstellt einen Screenshot und wechselt in den Annotationsmodus, falls das Programm noch nicht im Annotationsmodus ist. Außerdem wird nun Abschnitt B (C) sichtbar gemacht falls er noch nicht sichtbar war. Abschnitt A zeigt nur noch die nötigsten Informationen. Zentrales Element von Abschnitt B ist die Objektliste (D), in welcher alle erstellten Annotationsobjekte aufgeführt werden. Untermenüs werden rechts des Hauptfensters angezeigt (Markierung 5). Zurzeit besitzt nur das „Hinzufügen“ Werkzeug ein Untermenü, eines für GUI Elemente und eines für Webelemente. Um den Vorschlag abzusenden wechselt der Benutzer in Abschnitt C (E), wo er einen Titel und eine Beschreibung eingeben und den Namen des Absenders anpassen kann. Durch einen Mausklick auf den Knopf zum Abschicken (F) wird der Vorgang abgeschlossen.

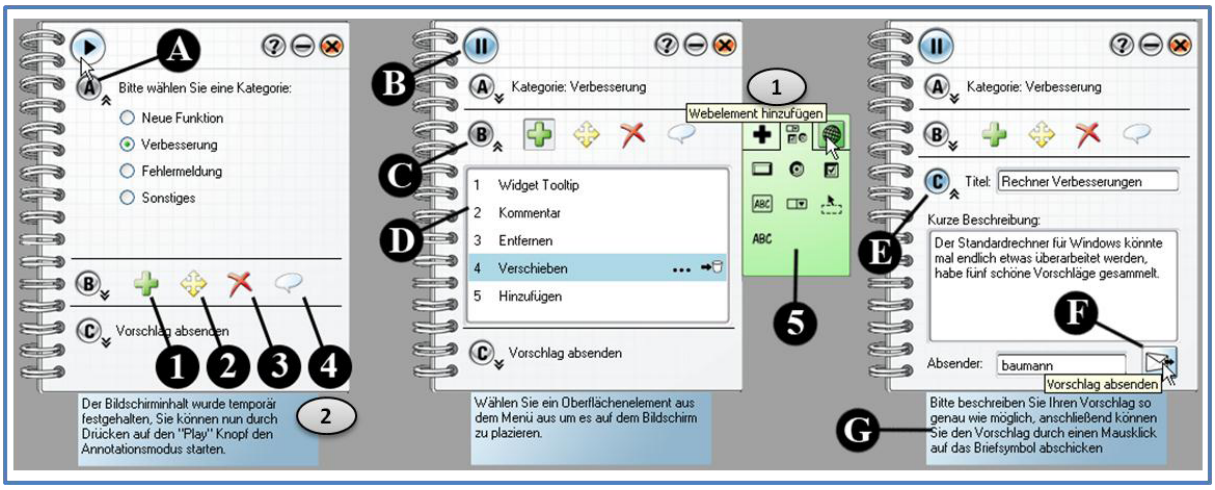

Abbildung 10.6: OpenProposal in Version 2.1

Die größte Änderung dieser Version stellte die Erweiterung (1) des „Hinzufügen“-Annotationswerkzeugs dar, welche auf Wunsch von Testpersonen erfolgte: Der Anwender hat die Möglichkeit vorgefertigte grafische Platzhalter, so genannte Widgets, auf dem Bildschirmfoto platzieren. Er kann direkt aus einem Angebot von typischen grafischen Softwareelementen (z.B. Schaltflächen, Checkbox, Radiobox, Beschriftung, Bild, Tabelle, etc.) auswählen und seinen Vorschlag somit konkretisieren. Beispielsweise kann mit dem Widget „Schaltfläche“ ausgedrückt werden, dass eine neue Schaltfläche gewünscht wird. Im Widget kann die Beschriftung der Schaltfläche sowie die gewünschte Funktionalität eingegeben werden. Allerdings zeigten die Untersuchungen, dass diese Funktionalität nur von versierten Benutzern genutzt und von Anwendern nicht zwingend benötigt wurde. Daher wurde diese Funktionalität in den späteren Versionen wieder entfernt.

Eine weitere Neuerung war eine permanent eingeblendete Hilfefunktion (2), die kontextsensitiv Hilfestellung anbietet. Während der Tests stellte sich heraus, dass diese Art der Hilfestellung keinen Mehrwert darstellte, da stattdessen der Moderator um Hilfe gefragt wurde.

Bereichert um die Erfahrungen der Fallstudie TRUMPF wurde Anfang 2008 die Version 2.43 fertiggestellt. Diese Version fand ihren Einsatz in der Fallstudie WAVES. Zahlreiche Funktionen wurden aufgrund der Anwenderbeobachtungen gestrichen bzw. automatisiert: Der Anwender brauchte seinen Vorschlag zu Beginn keiner Kategorie mehr zuzuordnen, da dies nun automatisiert aus der Annotation geschlossen wurde. Die kontextsensitive Hilfefunktion und die Objektliste wurden ausgeblendet, da diese nicht mehr benötigt wurden. Durch den neu gewonnen Platz auf der Oberfläche konnte die Darstellung vereinfacht werden. Ein Hauptmenu wurde implementiert, das über das OpenProposal-Logo (3) geöffnet werden konnte. Es enthielt Menu-Einträge zum Erstellen eines neuen Vorschlags, Wechseln in den Konfigurationsmodus, zum Erstellen eines neuen Vorschlags, zum Starten des Tutorials und zum Öffnen eines OpenProposal Vorschlags.

Zudem wurde das Angebot an Annotationswerkzeugen erweitert. Unter der Rubrik „Diskussion“ (1) standen die Annotationswerkzeuge „Frage zu einer Funktion“, „Fehler in einer Funktion“, „Lob zu einer Funktion“ und „Erweiterung einer Funktion“ zur Verfügung. Erstmals wurde die Schaltfläche „Bildschirmfoto anpassen“ (2) implementiert, mit der private Bereiche auf dem Bildschirm ausgeblendet werden konnten.

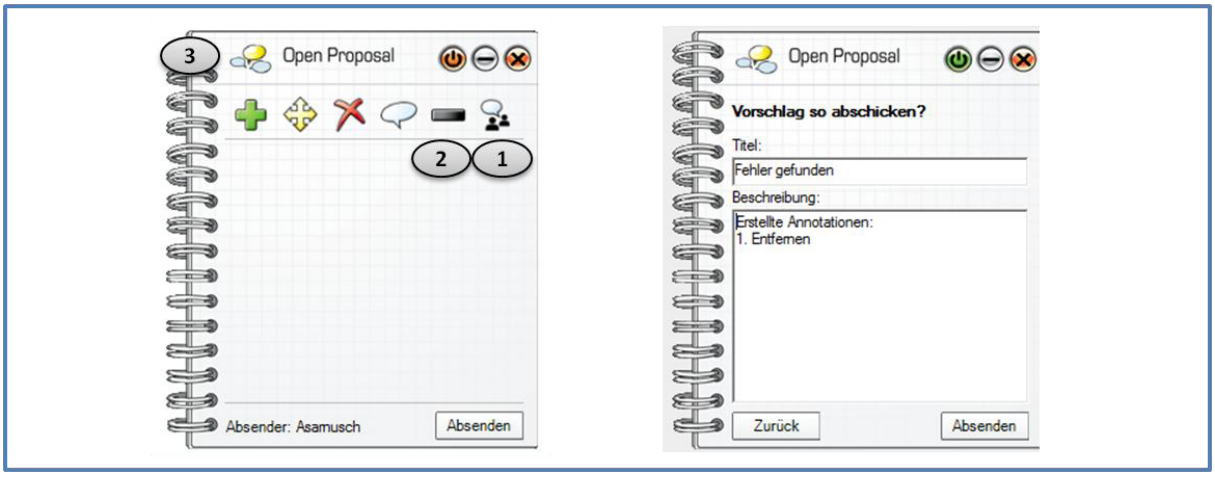

Abbildung 10.7: OpenProposal in Version 2.43

Im Jahr 2009 wurde aufbauend auf den Erfahrungen der Fallstudie WAVES und nachfolgenden Benutzertests die Version 2.5 fertiggestellt. Die Oberfläche wurde verkleinert, vereinfacht und auf das Wesentliche reduziert. Diese Version ist zum Zeitpunkt dieser Arbeit die aktuellste Version.

Abbildung 10.8: OpenProposal in Version 2.5