Eine Methode zur kollaborativen Anforderungserhebung und entscheidungsunterstützenden Anforderungsanalyse

Michael Geisser and Tobias Hildenbrand

Working Paper 6/2005
August 2005

Working Papers in Information Systems

University of Mannheim
Department of Information Systems 1
D-68131 Mannheim/Germany
Phone +49 621 1811691, Fax +49 621 1811692
E-Mail: wifo1@uni-mannheim.de
Internet: `http://www.bwl.uni-mannheim.de/wifo1`

„Eine Methode zur kollaborativen Anforderungserhebung und entscheidungsunterstützenden Anforderungsanalyse“

Michael Geisser und Tobias Hildenbrand

1 Einleitung

1.1 Motivation und Problemstellung

Bereits in den 60er Jahren wurden für die Softwareentwicklung ingenieurmäßige Methoden entwickelt, die unter der neu entstandenen Disziplin Software-Engineering (SE) subsumiert wurden. Deren Ziel ist es, zuverlässige Software zu entwickeln, welche sich durch gesicherte, hohe Qualität auszeichnet, kostengünstig innerhalb vorgegebener Budgetrahmen erstellt wird und zum geplanten Zeitpunkt bereitgestellt werden kann. Allerdings kommt eine aktuelle Umfrage in deutschen Firmen mit einem Umsatz von mehr als 250 Millionen Euro zu dem Ergebnis, dass 99% der SE-Projekte den ursprünglichen Zeitplan nicht einhalten können. Auch geschäftskritische Projekte, die in der Regel mit sehr hoher Priorität behandelt werden, überschritten zu 85% den geplanten Zeitpunkt der Auslieferung [Capgemini 2005].

Diese und weitere Probleme werden größtenteils durch eine unverstandene Problemstellung zu Beginn des Projekts sowie durch ein unsystematisches Vorgehen in den frühen Phasen der SE-Projekte verursacht [StandishGroup 1995, ESI 1996, StandishGroup 2001, Webster 2003]. Mit diesen frühen Phasen beschäftigt sich die Disziplin des *Requirements Engineering* (RE).

Durch ein systematischeres Vorgehen in der RE-Phase und eine Fokussierung ebendieser in Softwareentwicklungsprojekten sollen die Ziele des SE, nämlich zuverlässige und qualitativ hochwertige Software innerhalb planbarer monetärer und zeitlicher Budgets zu erstellen, erreicht werden. Dies wird erschwert, wenn es sich nicht nur um einen Auftraggeber, sondern um ein ganzes Konsortium von Auftraggebern mit einer Vielzahl von *Stakeholder* handelt. Gerade hier muss versucht werden, die verschiedenen Parteien systematisch zu einem Konsens im RE zu führen, auf dem die weiteren Phasen des SE aufbauen können.

1.2 Zielsetzung

Gegenstand dieses Beitrags ist die Entwicklung einer theoretisch fundierten Methode für eine kollaborative Anforderungserhebung mitsamt entscheidungsunterstützender Anforderungsanalyse.

Mit Hilfe dieser Methode und der passenden Werkzeugunterstützung sollen in erster Linie Systemhäuser in die Lage versetzt werden, als ersten Schritt des RE die Anforderungen eines

Konsortiums von Auftraggebern mit teilweise unterschiedlichen Interessen systematisch zu erheben. Zu diesem Zweck soll die Methode auf ein verteiltes Umfeld zugeschnitten sein, um eine mögliche geographische Verteilung der Auftraggeber zu berücksichtigen. Die Methode soll zudem systematische Entscheidungsunterstützung bei der Auswahl relevanter Anforderungen bieten, um so ein gerade bei verschiedenen Auftraggebern kritisches Maß an Objektivität zu gewährleisten und bei einem höheren Aufkommen an unterschiedlichen Anforderungen den Überblick zu bewahren.

1.3 Vorgehensweise und Gliederung

Nachdem die Problemstellung bereits aufgezeigt wurde, wird im zweiten Kapitel dieser Arbeit zunächst ein kurzer Überblick zu RE gegeben und bestehende kollaborative Ansätze im RE kritisch evaluiert. Der Fokus liegt hierbei auf einer kollaborativen Anforderungserhebung und -analyse, zudem wird diese Phase auch auf bestehende verteilte und quantitative Ansätze hin untersucht. Nach dieser Analyse wird im dritten Kapitel eine Methode für die kollaborative Anforderungserhebung und entscheidungsunterstützende Anforderungsanalyse entwickelt: Während zunächst die Anforderungen in einem verteilten Umfeld möglichst vollständig kollaborativ erhoben werden sollen, soll darauf aufbauend eine entscheidungsunterstützende Analyse die objektive Auswahl der zu implementierenden Anforderungen ermöglichen. Parallel zur Vorstellung der Methode wird auf die entsprechende Werkzeugunterstützung eingegangen Das vierte Kapitel fasst abschließend die Ergebnisse dieser Arbeit zusammen und gibt einen Ausblick auf zukünftige Forschungsfragen.

2 Existierende Ansätze im Requirements Engineering

Eingangs wurde aufgezeigt, dass die meisten Probleme in der Softwareentwicklung durch eine unverstandene Problemstellung hervorgerufen werden. Mit dieser Problematik beschäftigt sich die Disziplin des RE. An das RE schließen die Phasen der Analyse und des Entwurfs, der Implementierung sowie des Testens an und bilden so zusammen ein ganzheitliches SE-Projekt. Bis Anfang der 90er Jahre wurde noch nicht explizit von *Requirements Engineering* gesprochen, sondern der Begriff *Requirements Phase* verwendet, der sich aus den beiden Aktivitäten *Problemanalyse* sowie *Produktbeschreibung* zusammensetzt. Hierbei führt die *Problemanalyse* zu einem relativ vollständigen Verständnis der Anforderungen, während die anschließende *Produktbeschreibung* eine konsistente und vollständige *Software Requirements Specification* als Ergebnis hat [Davis 1990].

Eine häufig zitierte Definition des RE-Prozesses liefert Sommerville: „*The requirements for a system are the descriptions of the services provided by the system and its operational constraints. [...] The process of finding out, analysing, documenting and checking these services and constraints is called requirements engineering*“ [Sommerville, 2004, S.118].

Er unterteilt diesen Prozess in die vier Phasen

(1) Machbarkeitsstudie (*Feasibility study*),

(2) Anforderungserhebung und -analyse (*Requirements elicitation and analysis*),

(3) Anforderungsspezifikation (*Requirements specification*) und

(4) Anforderungsvalidierung (*Requirements validation*).

Zudem sieht Sommerville das Anforderungsmanagement (*Requirements Management*) als zusätzliche, nebenläufige Aktivität, die sich mit dem Management von Anforderungsänderungen beschäftigt [Sommerville 2004].

2.1 Kollaboratives RE

Die Softwareentwicklung an sich ist schon ein Prozess, der ohne Zusammenarbeit nur schwer oder gar nicht durchlaufen werden kann. So stellen Cook und Churcher fest: „*Software Engineering is inherently a team-based activity*“ [Cook and Churcher, 2003, S.290]. Als Teilgebiet des SE gilt diese Erkenntnis auch für das RE. Zudem ist gerade in den ersten Phasen eines Softwareentwicklungsprojekts die Einbeziehung aller für den Erfolg wichtigen Projektbeteiligten (*Stakeholder*) erfolgskritisch [Beyer and Holtzblatt 1995].

Die Anforderungserhebung und -analyse ist eine in höchstem Maße kollaborative Phase: *Stakeholder* werden zwischen Systemhaus und Auftraggeber identifiziert und Anforderungen werden daraufhin gemeinschaftlich unter Einbeziehung dieser *Stakeholder* erfasst. Das Analysieren der Anforderungen (Kategorisierung, Priorisierung und Auswahl) sowie die Bildung eines Konsenses finden vor allem zwischen den betroffenen *Stakeholder* auf Auftraggeberseite statt und werden in der Regel vom Systemhaus unterstützt und moderiert. Die Anforderungsspezifikation erfolgt auch kollaborativ: Die zentrale Aktivität dieser Phase, das Modellieren, kann nur dann erfolgreich sein, wenn während dieser Aktivität Rücksprachen mit den *Stakeholder* gehalten werden. Zahlreiche Autoren propagieren zudem eine stärkere Einbindung der *Stakeholder* in die Phase der Anforderungsspezifikation, beispielsweise durch Verwendung von Techniken wie *Joint Application Development* (JAD) oder *Rapid Application Development* (RAD) [Dean u.a. 1998, Graham 1998, Alexander 1999, Mahmood 2003].

Die anderen Phasen des RE sind weniger kollaborativ, sodass im Folgenden der Fokus auf bestehende kollaborative Ansätze in der Anforderungserhebung und –analyse gelegt werden soll.

2.2 Bestehende kollaborative Ansätze in der Anforderungserhebung und -analyse

In der Forschung hat sich im Bereich der kollaborativen Anforderungserhebung bisher nur der WinWin-Ansatz etabliert, der von der Theory W [Boehm und Ross 1989] ausgehend über die Erweiterung des Spiralmodells [Boehm und Bose 1994] bis hin zur mittlerweile vierten Generation des Ansatzes mit dem Namen EasyWinWin [Gruenbacher 2000] entwickelt wurde.

Die EasyWinWin-Methodik propagiert einen Wechsel von traditionellen, vertragsorientierten Mechanismen hin zu kooperativem Arbeiten mit Aufbau von Vertrauensbeziehungen. Mit der dadurch gewonnenen Flexibilität kann besser auf schnelle Technologiewechsel, Veränderungen in der Organisation sowie wechselnde *Stakeholder* reagiert werden. So werden keine rigiden Vertragswerke mit genau spezifizierten Anforderungen angestrebt, sondern versucht, die betreffenden *Stakeholder* mit einer gemeinsamen Vision und geteilten Vorstellungen auszustatten, um so bei unvorhergesehenen Problemen oder auch Möglichkeiten adaptiv und schnell zu reagieren [Boehm 2000, Highsmith 2000].

Von zentraler Bedeutung ist hierbei der Aufbau von Vertrauen innerhalb der Arbeitsgruppe. Die EasyWinWin-Methodik führt zudem zu realistischeren Erwartungen, da durch einen intensiven Diskurs die jeweiligen Vorstellungen ausgetauscht und kritisch hinterfragt werden können. Gruenbacher und Briggs zeigen zudem, dass mit Hilfe der EasyWinWin-Methodik verborgenes Wissen (*tacit knowledge*) aufgedeckt, sowie Konflikte und Widersprüche sehr früh erkannt werden können [Gruenbacher und Briggs 2001]. Ein weiterer Vorteil liegt in einer detaillierten Prozessbeschreibung [Gruenbacher 2002], die das Team sicher durch den Prozess führt sowie in der Prozessunterstützung mittels Groupware: es wurde eine spezielle Groupware entwickelt, die den Prozess mittlerweile so gut unterstützt, dass diese Unterstützung als integraler Bestandteil der Methodik angesehen wird und ebenfalls den Namen EasyWinWin trägt: „*EasyWinWin combines the WinWin Spiral Model of Software Engineering with collaborative knowledge techniques and automation of a Group Support System*“ [Briggs und Gruenbacher, 2002, S.4].

Als nachteilig wirkt sich die relativ hohe Komplexität der EasyWinWin-Methodik aus. So ist der Prozess nicht sehr intuitiv und wird nur mit einer Schulung des Moderators und der Teilnehmer Erfolg versprechend sein. Zudem ist der Prozess nicht auf ein verteiltes Umfeld zuge-

schnitten, da traditionelle Diskussionen einen wichtigen Baustein der Methodik bilden. Ein weiterer Nachteil ist die relativ hohe Subjektivität bei der Auswahl der Anforderungen. Es wird zwar versucht, ein gewisses Maß an Objektivität zu erzeugen, doch wird hierfür zum einen eine absolute Priorisierung und keine vergleichende verwendet[1], zum anderen wird zwar die Einfachheit in der Umsetzung (*Ease of Realization*) beurteilt, allerdings fließen in dieses Kriterium sehr viele Faktoren (technologische, soziale, politische und ökonomische) ein. Es ist daher zweifelhaft, ob die teilnehmenden *Stakeholder* in der Lage sind, ein solch komplexes Kriterium auf einer absoluten Skala genau zu bewerten. Auch die Anweisung, nur dann abzustimmen, wenn ein *Stakeholder* meint, das Kriterium einschätzen zu können, kann kritisch gesehen werden, da die subjektive Einschätzung signifikant von der tatsächlichen Fähigkeit abweichen kann. Für eine Übersicht der Vor- und Nachteile der EasyWinWin-Methodik siehe Tabelle 1.

Vorteile	Nachteile
(+) Flexibilität	(-) Nicht sehr intuitiv
(+) Aufbau von Vertrauen	(-) Nicht für ein verteiltes Umfeld konzipiert
(+) Realistische Erwartungen	(-) Relativ hohe Subjektivität
(+) Aufdeckung von verborgenem Wissen	
(+) Frühe Erkennung von Konflikten	
(+) Detaillierte Prozessbeschreibung	
(+) Unterstützung durch Groupware	

Tabelle 1: Vor- und Nachteile der EasyWinWin-Methodik

2.3 Anforderungserhebung und -analyse in einem verteilten Umfeld

Durch die fortschreitende Globalisierung der Wirtschaft und eine damit verbundene Zunahme an globalen, strategischen Partnerschaften, internationalen Firmenzusammenschlüssen in Joint Ventures und global aufgestellten Weltkonzernen wird auch ein verteiltes Entwickeln von Software (*Global Software Development*) unumgänglich [Karolak 1998]. Der treibende Faktor aus organisatorischer Sicht ist hierbei die Möglichkeit, Ressourcen, wie bspw. Hardware und Expertenwissen, gemeinsam zu nutzen [French und Layzel 1998]. Ermöglicht wur-

[1] Nach Karlsson und Ryan führt eine absolute Priorisierung zu schlechteren Ergebnissen als eine vergleichende Priorisierung [Karlsson und Ryan 1997].

de ein verteilter Entwicklungsprozess in den letzten Jahren durch technologischen Fortschritt, der eine ausreichende Kommunikationsinfrastruktur, Bandbreite sowie Leistung mit sich brachte.

Vor diesem Hintergrund wurde neben einer verteilten Softwareentwicklung [Herbsleb et al. 2001] auch explizit der Prozess des verteilten RE, mit Fokus auf die Anforderungserhebung, empirisch untersucht [Damian et al. 2003, Damian 2001, Llyod et al. 2002, Ocker et al. 1995]. Diese Studien haben gemeinsam, dass sie eine verteilte Anforderungserhebung unter Einsatz von traditionellen Techniken und ohne Verwendung einer auf dieses verteilte Umfeld angepassten Methode untersuchen. Zudem werden einige asynchrone Techniken, wie das gemeinsame Aufstellen eines Glossars oder die Verwendung von Foren für die Anforderungserhebung, nicht berücksichtigt. Trotzdem kommen sämtliche Studien zu dem Ergebnis, dass eine verteilte Anforderungserhebung möglich und auch vorteilhaft sein kann.

Um die Vorteilhaftigkeit einer verteilten Anforderungserhebung zu verwirklichen, müssen sowohl methodische Prinzipien berücksichtigt als auch Anforderungen an eine Kollaborationssoftware gestellt werden. Für Ersteres stellen Damian u.a. Richtlinien auf, welche die Vorteilhaftigkeit einer verteilten Anforderungserhebung sicherstellen sollen [Damian u.a. 2003]. So sollen z.B. unbedingt anfangs persönliche Treffen stattfinden, um einen Vertrauensaufbau bei den beteiligten Personen zu ermöglichen. Anforderungen an die Groupware umfassen unter anderem die Unterstützung von synchroner und asynchroner Zusammenarbeit [Herlea und Greenberg 1998].

Da geographisch entfernte *Stakeholder* mit der EasyWinWin-Groupware lediglich rudimentär eingebunden werden können, wurde von Gruenbacher und Braunsberger ein webbasiertes Werkzeug zur Anforderungserhebung entwickelt, das die EasyWinWin-Methodik unterstützt: ARENA (Anytime, Anyplace REquirements Negotiation Aids) [Gruenbacher und Braunsberger 2003]. Dieses Werkzeug ergänzt allerdings nicht die bestehende Groupware, sondern ersetzt diese. In einem verteilten Umfeld muss somit der gesamte Prozess in ARENA stattfinden. Dies ist problematisch, da ARENA ausschließlich webbasiertes, asynchrones Arbeiten unterstützt. Somit können keine synchronen Treffen unterstützt werden, die gerade bei der EasyWinWin-Methodik eine kritische Rolle einnehmen. Weiterhin problematisch ist die nicht intuitive Bedienung, auf die sich zudem das physische Fehlen eines Moderators negativ auswirkt.

Neben ARENA wurden zwei mobile Werkzeuge entwickelt, mit denen Anforderungen auf mobilen Endgeräten auf Basis der EasyWinWin-Methodik erhoben werden können [Seyff u.a.

2004]. Diese mobilen Werkzeuge sind vor allem dann nützlich, wenn die mit der Ermittlung der Anforderungen betrauten Personen zusätzlich zu den EasyWinWin-Workshops Interviews mit geographisch verteilten *Stakeholder* komplementär durchführen wollen.

Die Entwicklung von Open Source-Software findet fast ausschließlich verteilt statt. Daher sind insbesondere die hierbei eingesetzten Methoden der Anforderungserhebung zu analysieren. Dies kann weitere Anhaltspunkte liefern, wie eine Methode für die Anforderungserhebung intuitiver als im EasyWinWin-Prozess und vollständig verteilt ablaufen kann.

Die Hauptunterschiede zwischen kommerzieller Softwareentwicklung und *Open Source Software Development* (OSSD) liegen in den Anforderungsprozessen, da im Gegensatz zur kommerziellen Softwareentwicklung im OSSD die Entwickler auch zu den späteren Nutzern der Software gehören [Massey 2002]. Empirische Untersuchungen zeigen, dass die Anforderungsprozesse im OSSD sowohl implizit als auch informell verlaufen und einige Aktivitäten des RE überhaupt nicht abgedeckt werden [Massey 2002,2003, Scacchi 2002, Scacchi u.a. 2004]. Die Anforderungserhebung und -analyse erfolgt hierbei informeller als im RE, indem die Anforderungen in Foren und über Mailinglisten erhoben und diskutiert werden. Gerade im Falle eines verteilten Umfelds stellen Foren eine Möglichkeit dar, Anforderungen auch in der kommerziellen Softwareentwicklung ressourcensparend zu erheben. Diese Foren sollten aber strukturiert und von einem Moderator betreut werden, um die mit diesem Medium vielleicht weniger vertrauten *Stakeholder* anzuleiten und den Prozess möglichst systematisch ablaufen zu lassen.

2.4 Quantitative Ansätze

Viele Methodiken im RE, wie auch die EasyWinWin-Methodik, beruhen auf subjektiven Einschätzungen, so dass gerade hier eine mit dem Einsatz von *Decision Support* verbundene objektivere Betrachtung zu besseren Entscheidungen führen kann. Im RE muss auf die Wünsche verschiedenster *Stakeholder* eingegangen werden, die verschiedene Perspektiven auf das geplante System und somit unterschiedliche Prioritäten haben können. Zudem ist es vielen beteiligten Personen unklar, wie aufwendig sich die Implementation einzelner Anforderungen gestalten wird. Aufgrund von Budgetbeschränkungen und Beachtung der Produkteinführungszeit (*time to market*) können aber in der Regel nicht alle Wünsche der *Stakeholder* im geplanten System berücksichtigt werden.

Daher gilt es, aus allen zur Disposition stehenden Anforderungen diejenigen auszuwählen, die unter Berücksichtigung aller Budgetbeschränkungen sowie der Produkteinführungszeit den

Nutzen und somit den Wert für den Kunden maximieren [Ruhe u.a. 2003]. In der Literatur finden sich vor allem zwei quantitative Methoden, die eine solche Auswahl unterstützen: der *Cost-Value Approach* [Karlsson und Ryan 1997] und *Quantitative WinWin* [Ruhe u. a. 2003]. Beide Methoden basieren auf dem *Analytic Hierarchy Process* [Saaty, 1980], der sich als Priorisierungsalgorithmus für Anforderungen im Softwarebereich im Vergleich mit anderen Ansätzen als überlegen erwiesen hat [Karlsson u.a. 1998].

Die Vorteile des ***Cost-Value Approach*** bestehen darin, dass er intuitiv und für die Nutzer leicht zu bedienen ist. Darüber hinaus liefert er durch sein mathematisches Fundament bessere Ergebnisse als absolute Beurteilungen, welche z.B. innerhalb der EasyWinWin-Methodik ihre Anwendung finden. Nachteilig auf den *Cost-Value Approach* wirken sich die paarweisen Vergleiche des AHP aus, welche die Komplexität mit zunehmender Anzahl von Anforderungen stark ansteigen lassen. Auch Abhängigkeiten zwischen Anforderungen werden nicht berücksichtigt. So kann z.B. eine Anforderung mit einem sehr niedrigen Nutzen-Kosten-Verhältnis für die Implementierung einer anderen Anforderung mit einem sehr hohen Nutzen-Kosten-Verhältnis unentbehrlich sein, der *Cost-Value Approach* würde aber dazu raten, diese unentbehrliche Anforderung nicht zu implementieren.

Die Vorteile von ***Quantitative WinWin*** bestehen darin, dass er wie der *Cost-Value Approach* ein solides mathematisches Fundament besitzt und daher geeignet ist, eine wesentliche Limitation der EasyWinWin-Methodik, nämlich die Subjektivität, zu überwinden. Zudem werden die Anzahl der Vergleiche durch Einsatz eines hierarchischen AHP deutlich reduziert, ein beträchtlicher Fortschritt gegenüber dem *Cost-Value Approach*. Allerdings wirken sich nach wie vor die paarweisen Vergleiche des AHP nachteilig aus, da gerade bei einem solchen iterativen Ansatz (und damit einhergehend ein mehrmaliger Durchlauf des AHP) eine hohe Kooperation und Teilnahmebereitschaft von Seiten der *Stakeholder* erforderlich ist. Diese Kooperation wird durch die Bewertung der relativen Wichtigkeiten von Anforderungen in der Erweiterung von *Quantitative WinWin* [Ruhe u. a. 2003] noch erfolgskritischer, allerdings ist die Annahme der ursprünglichen Methode, dass diese Wichtigkeiten gegeben sind, sehr realitätsfern und vereinfachend. Ein großes Risiko, stellt die Schätzung des Aufwands (sowie der Dauer und der Qualität in der erweiterten Methode) mittels des Simulationssystems GENSIM von Pfahl dar [Ruhe u. a. 2003, Pfahl 2001]. Dass eine solch genaue Schätzung pro Anforderung zuverlässige Ergebnisse liefert, darf zumindest bezweifelt werden. Ferner werden in der Methode weder Konsistenztests der AHP-Ergebnisse durchgeführt, noch Abhängigkeiten von Anforderungen berücksichtigt. Diese Abhängigkeiten sind auch deshalb wichtig, da davon

ausgegangen werden kann, dass sich sowohl der Nutzen als auch die Komplexität einzelner Anforderungen nicht konstant verhält, sondern mit steigender Anzahl an implementierten Anforderungen zunimmt [Ruhe 2003]. Irreführend an sich ist auch der Name der Methode, da bis auf die iterative Natur des Prozesses keine weiteren Bestandteile des WinWin-Ansatzes Verwendung finden. Tabelle 2 gibt einen Überblick über die Vor- und Nachteile der analysierten Methoden.

Methode	**Vorteile**	**Nachteile**
Cost-Value Approach	(+) Mathematisches Fundament (+) Kosten-Nutzen-Abwägungen (+) Konsistenz-Test (+) Intuitiv zu bedienen	(-) Abhängigkeiten von Anforderungen nicht berücksichtigt (-) Hohe Komplexität
Quantitative WinWin	(+) Mathematisches Fundament (+) Kosten-Nutzen-Abwägungen (+) Hierarchischer Einsatz des AHP	(-) Abhängigkeiten von Anforderungen nicht berücksichtigt (-) Hohe Komplexität (-) Kein Konsistenz-Test (-) Aufwandsschätzung problematisch (-) Enge Kooperation der *Stakeholder* benötigt

Tabelle 2: Gegenüberstellung von *Cost-Value Approach* und *Quantitative WinWin*

3. Entwicklung der Methode

Ausgehend von den Erkenntnissen der existierenden Ansätze soll nun eine Methode für die kollaborative Anforderungserhebung in einem verteilten Umfeld vorgestellt werden, die durch Einbeziehung quantitativer Techniken Entscheidungsunterstützung bei der Auswahl von Anforderungen bietet. Diese Methode besteht aus zwei Phasen: zuerst werden die Anforderungen ähnlich der EasyWinWin-Methodik iterativ erfasst, allerdings mit expliziter Berücksichtigung von verteiltem Arbeiten. Danach erfolgt eine Analyse der Kosten und des Nutzens, um die Auswahl der zu implementierenden Anforderungen zu unterstützen. Durch diese Zweiteilung kann sichergestellt werden, dass zunächst sämtliche Anforderungen aufgedeckt und Beschränkungen durch Budgetrestriktionen erst bei der Auswahl der zu implementierenden Anforderungen berücksichtigt werden. Mit Hilfe dieser Methode sollen Systemhäuser in die Lage versetzt werden, als ersten Schritt des RE die Anforderungen der verschiedenen *Stakehol-*

der des Auftraggebers oder auch eines Konsortiums von Auftraggebern mit teilweise verschiedenen Interessen systematisch zu erheben.

3.1 Kollaborative Anforderungserhebung

In der ersten Phase der Methode werden die Anforderungen kollaborativ und iterativ erhoben. Die Methode baut hier auf den Erkenntnissen der EasyWinWin-Methodik auf, verwendet aber Techniken aus dem Open Source-Umfeld, um das Vorgehen intuitiver zu gestalten. Zudem ist sie im Gegensatz zu EasyWinWin durchgängig für ein verteiltes Umfeld konzipiert. Ziel dieser ersten Phase ist es, die funktionalen Anforderungen möglichst vollständig zu erfassen.

Als Input dient eine vage Vision, in der die Wünsche an die Software des Auftraggebers formuliert sind. Zudem liegt in der Regel eine erste Liste mit *Stakeholder* vor, die Anforderungen an das System stellen. Der in den folgenden Abschnitten vorgestellte Prozess soll ein Systemhaus dabei unterstützen, die Anforderungen an die geplante Software, von einer anfänglichen Vision ausgehend, durch Einbeziehung aller involvierten *Stakeholder*[2] systematisch und möglichst vollständig zu erheben. Die Entwicklung dieser Vision stellt dabei einen zentralen Punkt dar. Die einzelnen Schritte dieser ersten Phase der Methode werden im Folgenden detailliert beschrieben. Abbildung 1 zeigt eine Übersicht der Schritte in Form einer Spirale, die den iterativen Charakter der Methode visualisieren soll.

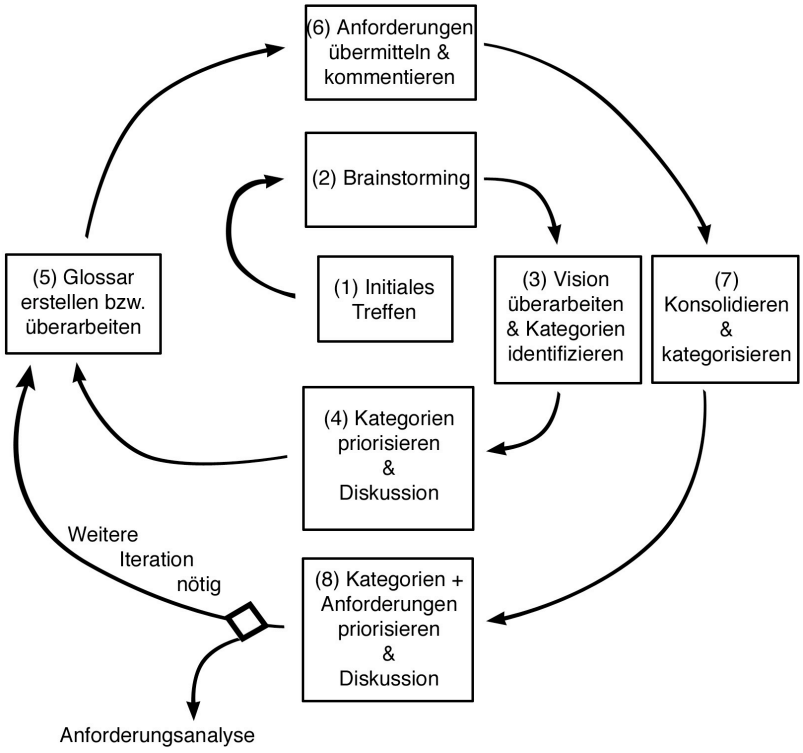


Abbildung 1: Kollaborative Anforderungserhebung

[2] Die Zusammensetzung der Stakeholder kann hierbei über die Iterationen evolvieren.

1. Schritt: Anfängliches Treffen

Bei diesem ersten Treffen wird die Vision mitsamt einer ersten Liste von *Stakeholder* an das Systemhaus übergeben und der folgende Prozess besprochen. Dabei sollen noch keine Anforderungen erhoben oder neue Ideen generiert werden, sondern maximal die Vision etwas präzisiert werden, falls dies für nötig befunden wird. Bevor in den nächsten Schritten weitestgehend auf asynchrones, verteiltes Arbeiten gesetzt wird, können sich bei diesem Treffen alle Beteiligten persönlich kennen lernen.

2. Schritt: Asynchrones Brainstorming

Es findet zunächst ein asynchrones Brainstorming statt, um die ersten Ideen zu generieren. Dies soll in einem webbasierten Forum stattfinden, um ein verteiltes Arbeiten zu ermöglichen. Hierbei können alle *Stakeholder* neue Ideen generieren sowie andere Einträge ergänzen und kommentieren. Während bei dem Brainstorming der EasyWinWin-Methodik noch keine Kritik geübt werden soll, wird dies hier ausdrücklich verlangt, um möglichst schnell unrealistische Anforderungen auszusortieren. In diesem Schritt werden somit die zwei Phasen eines klassischen Brainstormings, nämlich die Generierung von Ideen und die anschließende Analyse mit Reduktion [Leffingwell und Widrig 2000], in einem Schritt interferierend bewältigt.

Es sollen hier noch keine detaillierten Anforderungen diskutiert werden, sondern nur Ideen und Vorschläge generiert werden, aus denen sich in späteren Phasen konkrete und detailliertere Anforderungen ableiten lassen. Um den Prozess zu unterstützen, stellt das Systemhaus einen Moderator, der intensiv an diesem Schritt teilnimmt, indem er überwacht, dass die Diskussionen nicht zu detailliert werden. Zudem soll er bei Unklarheiten und bezüglich der zu verwendeten Fachtermini Fragen stellen und versuchen, die aktive Teilnahme aller Beteiligten zu sichern.

3. Schritt: Vision überarbeiten und Kategorien identifizieren

Nach Abschluss der Brainstorming-Phase wird die Vision durch den Moderator sowie einen weiteren IT-Experten des Systemhauses überarbeitet, indem die aus dem vorherigen Schritt generierten Ideen eingearbeitet werden. Es werden zudem Kategorien von Anforderungen aus den übermittelten Ideen gebildet, um so in den nächsten Schritten ein strukturiertes Vorgehen zu erlauben. Der IT-Experte identifiziert dabei unrealistische Vorschläge und stellt somit die Realisierbarkeit des Projekts sicher. Zudem sollen Fachtermini identifiziert werden, deren Bedeutung in einem Glossar definiert werden muss.

4. Schritt: Priorisierung der Kategorien mit Diskussion

Die Priorisierung der Kategorien sowie die anschließende Diskussion finden als virtuelles Treffen statt. Neben dem Moderator, der alle Beteiligten durch den Prozess führt, und allen *Stakeholder* auf Kundenseite, nimmt auch der IT-Experte aus dem vorigen Schritt an diesem Treffen teil. Um dieses virtuelle Treffen zu ermöglichen, ist eine Groupware nötig, die Video- und Audioübertragungen zulässt und über die Möglichkeit verfügt, anonyme Abstimmungen durchzuführen[3].

Zunächst werden die unrealistischen Vorschläge, die im vorigen Schritt gestrichen wurden, durch den IT-Experten des Systemhauses genannt und die Gründe erläutert. Nun findet zunächst eine anonyme Priorisierung der Kategorien von Seiten der *Stakeholder* statt. Dabei soll die Wichtigkeit der Kategorien aus Sicht der Unternehmung, unter Verwendung einer Skala von 0 (überhaupt nicht wichtig) bis 3 (überaus wichtig), beurteilt werden. Eine feinere Einteilung der Skala ist hier nicht dienlich, da die Vorstellungen noch nicht sehr konkret sind sowie nur signifikante Abweichungen erkannt werden sollen. Weichen die Einschätzungen der *Stakeholder* zu einer Kategorie stark von einander ab (d.h. um mehr als 1 Punkt auf der Skala), so soll über die Bedeutung der Kategorie intensiv diskutiert werden.

Ziel dieser Diskussion ist es, einen Konsens unter den beteiligten *Stakeholder* zu erreichen. Weichen die Einschätzungen einer Kategorie nicht stark voneinander ab, so soll die Bedeutung durch den Moderator kurz zusammengefasst und nur bei Bedarf diskutiert werden, z.B. wenn ein Beteiligter eine Frage hierzu hat. Nach der Diskussion wird die Vision präsentiert und bei Bedarf verändert. Nun soll die Liste mit den im vorigen Schritt identifizierten Begriffen aus dem Glossar vorgestellt werden und falls nötig um neue Fachtermini ergänzt werden.

Abschließend wird entschieden, ob neue *Stakeholder* in den weiteren Verlauf der Anforderungserhebung einbezogen werden sollen und ob jetzige *Stakeholder* für den weiteren Prozess der Anforderungserhebung zunächst nicht mehr benötigt werden.

5. Schritt: Asynchrone Erstellung eines Glossars

Die Erstellung eines Glossars für die im vorigen Schritt identifizierten Fachtermini soll asynchron mittels einer webbasierten Technik erfolgen. Denkbar wäre z.B. der Einsatz eines Wiki-Systems [Leuf und Cunningham 2001] oder anderer Groupware, die eine gemeinsame, asynchrone Erstellung eines Glossars webbasiert erlaubt. Dabei soll vor allem auf die Abgrenzung

[3] beispielsweise spreed.com der Firma struktur AG: http://www.spreed.com/

der Fachtermini untereinander geachtet, und auf synonymische Verwendungen von Begriffen hingewiesen werden.

6. Schritt: Anforderungen übermitteln und kommentieren

Um die Anforderungen asynchron und verteilt zu übermitteln sowie bereits übermittelte Anforderungen zu kommentieren, dient ein strukturiertes, webbasiertes Forum. Der Moderator erstellt in einem solchen Forum pro Kategorie einen Bereich sowie einen zusätzlichen Bereich für Anforderungen, die keiner bestehenden Kategorie zugeordnet werden können.

Daraufhin erstellen die *Stakeholder* zu jeder neu übermittelten Anforderung einen Diskussionsfaden (*Thread*) und kommentieren bereits bestehende *Threads*. Auch in diesem Schritt stellt der Moderator bei Unklarheiten Fragen, bittet bei Bedarf um nähere Erläuterungen und versucht, die aktive Teilnahme aller Beteiligten zu sichern (vgl. Schritt 2).

7. Schritt: Anforderungen konsolidieren und kategorisieren

Nun werden die in den Forumsdiskussionen entstandenen Anforderungen durch den Moderator und den IT-Experten konsolidiert, d.h. alle Erkenntnisse eines Diskussionsfadens werden zusammengefasst. Dann werden diese konsolidierten Anforderungen bestehenden Kategorien oder neu zu erstellenden Kategorien zugeordnet. Bei der Zuordnung der Anforderungen zu den Kategorien achtet der IT-Experte darauf, dass untereinander abhängige Anforderungen nicht in verschiedene Kategorien eingeordnet werden.

Falls nötig, müssen hierzu die bestehenden Kategorien reorganisiert werden. Wie schon im dritten Schritt werden dabei unrealistische Vorschläge durch den IT-Experten identifiziert und Fachtermini aufgelistet, die in das Glossar aufgenommen werden müssen. Bei Bedarf kann auch die Vision überarbeitet werden.

8. Schritt: Priorisierung der Kategorien und Anforderungen mit Diskussion

Um sowohl kosten- und zeiteffektiv arbeiten zu können, als auch Vertrauen und persönliche Beziehungen im Team aufzubauen, sollen sich physische und virtuelle Treffen abwechseln. Wird dieser Schritt also iteriert und fand das letzte Treffen physisch statt, so wird in der aktuellen Iteration ein virtuelles Treffen erfolgen. Dieser achte Schritt erfolgt analog zum vierten Schritt, allerdings werden neben den Kategorien auch die Anforderungen priorisiert und diskutiert.

Zudem wird abschließend entschieden, ob eine weitere Iteration vonnöten ist. Muss das Glossar überarbeitet werden oder sind neue *Stakeholder* identifiziert, muss auf jeden Fall eine

weitere Iteration ab Schritt fünf erfolgen. Ansonsten müssen die Beteiligten prüfen, ob die Kategorien vollständig und innerhalb jeder Kategorie alle in Frage kommenden Anforderungen vorhanden sind. Ist dies noch nicht der Fall, so findet eine (Teil-)Iteration statt, indem die Schritte fünf bis acht wiederholt werden. Wird entschieden, dass keine weitere Iteration nötig ist, so ist die Phase der Anforderungserhebung abgeschlossen.

3.2 Entscheidungsunterstützende Anforderungsanalyse

In der zweiten Phase der Methode werden die zu implementierenden Anforderungen auf der Basis einer quantitativen Analyse von Kosten und Nutzen ausgewählt. Ausgangspunkt ist eine Liste mit Anforderungen an ein geplantes System. Aus ökonomischen Gesichtspunkten ist es vernünftig, nur solche Anforderungen zu realisieren, deren Kosten in der Implementation durch einen ausreichend hohen Nutzen gerechtfertigt werden. Auch monetäre Budgetbeschränkungen können eine Auswahl von Anforderungen zwingend erfordern, wobei diese am besten so zu erfolgen hat, dass die Anforderungen mit den größten Nutzen-Kosten-Verhältnissen implementiert werden. Die vier Schritte dieser Phase werden im Folgenden detailliert beschrieben. Für eine Übersicht sei auf Abbildung 2 verwiesen.

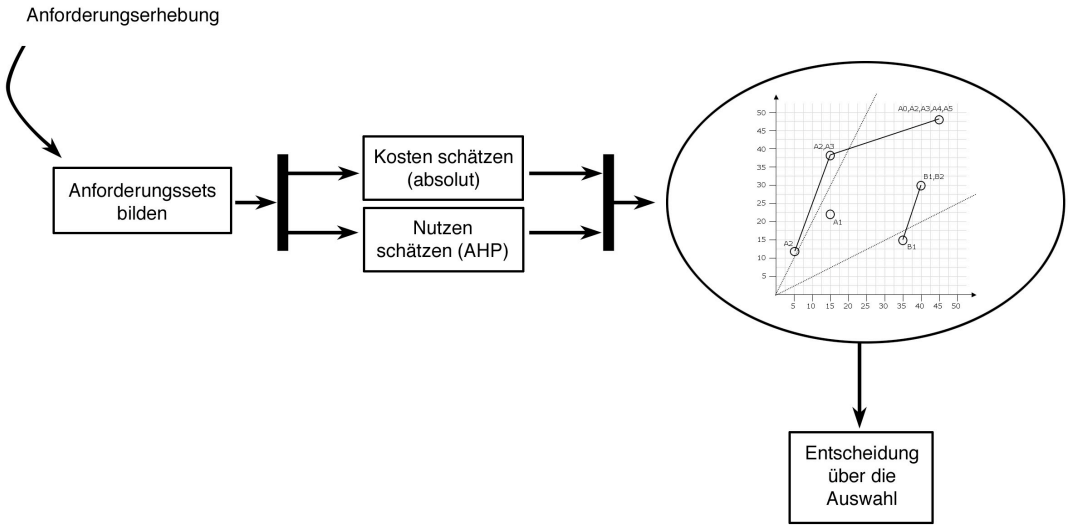


Abbildung 2: Entscheidungsunterstützende Anforderungsanalyse

1. Schritt: Anforderungssets bilden

Da Anforderungen untereinander Abhängigkeiten aufweisen können, dürfen sie nicht unter Vernachlässigung dieser Abhängigkeiten miteinander verglichen werden. Enthält eine Kategorie voneinander abhängige Anforderungen, so werden für diese jeweils so genannte „Anforderungssets“ gebildet, die in einer entsprechenden Applikation mittels eines gerichteten Graphen grafisch dargestellt werden können. Zur Anschauung sei auf Abbildung 3 verwiesen:

Hier ist Anforderung A2 Voraussetzung für A3, letzte wiederum zusammen mit A0 Voraussetzung für A4 und A5. Zusammen bilden sie ein abgeschlossenes Anforderungsset.

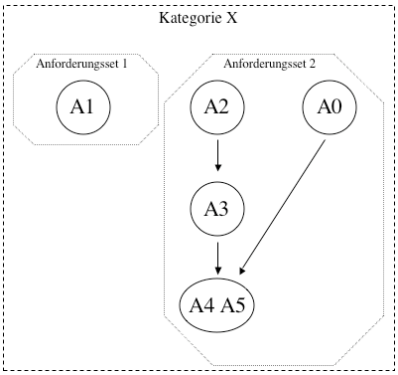


Abbildung 3: Darstellung von Abhängigkeiten mittels gerichteten Graphen

2. Schritt: Kosten und Nutzen schätzen

Sind innerhalb der Kategorien die Anforderungssets gebildet, so können die Kosten und der Nutzen der Anforderungen geschätzt werden. Während es den IT-Experten des Systemhauses obliegt, realistische Kostenschätzungen für die Anforderungen zu geben, sind ausschließlich die *Stakeholder* auf Kundenseite für die Schätzung des Nutzens verantwortlich. Während die Kosten unter Einbeziehung einer Mengenkomponente (z.B. Mannmonate oder -tage) und einer Wertkomponente (Tagessatz je Mitarbeiter) geschätzt werden, wird der Nutzen mittels hierarchischen Einsatzes des AHP bestimmt.

3. Schritt: Grafische Darstellung der Ergebnisse

Die Ergebnisse des zweiten Schritts werden unter Berücksichtigung der Abhängigkeiten grafisch dargestellt, indem vom Wurzelelement der gerichteten Graphen ausgehend alle möglichen Kombinationen mit ihren aggregierten Nutzenwerten und Kosten in ein Kosten-Nutzen-Diagramm eingetragen werden (siehe Abbildung 4, Fortführung des Beispiels aus Abb. 3).

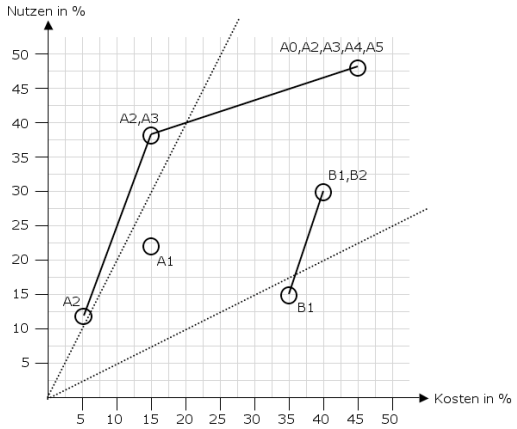


Abbildung 4: Kosten-Nutzen-Diagramm

Um die Schätzung der Kosten und des Nutzen zu unterstützen, wurde an der Universität Mannheim eine prototypische webbasierte Anwendung entwickelt. Sie ermöglicht es, die *Stakeholder* sicher durch den Prozess zu führen und die Nutzenwerte mit Hilfe des AHP zu berechnen und grafisch in einem Kosten-Nutzen-Diagramm auszugeben (siehe Anhang 1).

4. Schritt: Entscheidung über die Auswahl

Abschließend findet ein physisches Treffen aller *Stakeholder* statt, bei dem der Moderator das Kosten-Nutzen-Diagramm mit den eingetragenen Anforderungen und Anforderungssets präsentiert. Auf Basis dieser objektivierten Entscheidungsgrundlage wird nun diskutiert und entschieden, welche Anforderungen implementiert, und welche Anforderungen verworfen bzw. für eine spätere Version des Produkts vorgehalten werden.

Um die Entscheidung zu vereinfachen, enthält das Kosten-Nutzen-Diagramm zwei Geraden: Anforderungen, die oberhalb der Gerade mit $y = 2x$ liegen, sollten auf jeden Fall implementiert werden, während Anforderungen, die unterhalb der Gerade $y = 1/2\ x$ liegen, nicht berücksichtigt werden sollten (siehe Abbildung 4). Die Einteilung des Diagramms mittels der zwei Geraden hat sich empirisch bewährt, um Anforderungen mit einem hohen Nutzen-Kosten-Verhältnis von solchen mit einem niedrigen abzugrenzen [Karlsson und Ryan 1997].

4. Zusammenfassung und Ausblick

In diesem Beitrag wurde, aufbauend auf einer kritischen Evaluation bestehender Ansätze, eine entscheidungsunterstützende Methode für eine kollaborative Anforderungserhebung und -analyse in einem verteilten Umfeld entwickelt. Diese setzt sich aus zwei aufeinander folgenden Phasen zusammen. Während in der ersten Phase die Anforderungen iterativ und möglichst vollständig erhoben werden, erfolgt in der zweiten Phase die Auswahl der tatsächlich zu implementierenden Anforderungen (siehe Anhang 2).

Mit Hilfe dieser Methode kann für eine kollaborative Anforderungserhebung ein systematischeres Vorgehen während der RE-Phase und eine stärkere Fokussierung ebendieser in verteilten Softwareentwicklungsprojekten ermöglicht werden. Dies bedeutet einen weiteren Schritt in Richtung des Ziels des SE, nämlich zuverlässige und qualitativ hochwertige Software innerhalb planbarer monetärer und zeitlicher Budgets zu erstellen.

Systemhäuser werden somit in die Lage versetzt, als ersten Schritt des RE die Anforderungen eines Konsortiums von Auftraggebern mit teilweise verschiedenen Interessen systematisch zu erheben. Dies wird erreicht, indem die Methode den bereits etablierten WinWin-Ansatz auf

ein verteiltes Umfeld überträgt und ihn insgesamt intuitiver sowie bei der Auswahl von Anforderungen objektiver gestaltet.

Durch diese Vorgehensweise kann auf die zahlreichen theoretischen und empirischen Erkenntnisse des WinWin-Ansatzes zurückgegriffen werden. Durch Analyse bestehender Praktiken in der verteilten Softwareentwicklung, insbesondere im Open Source-Umfeld, sowie durch Einbeziehung quantitativer Methoden konnten die bekannten Schwachstellen der EasyWinWin-Methodik behoben werden. Neben Vermeidung der Nachteile des WinWin-Ansatzes ist es zudem zum ersten Mal gelungen, Abhängigkeiten zwischen Anforderungen in Form von Sets systematisch bei der Anforderungsanalyse zu berücksichtigen.

Um die Entwicklung der Methode abzuschließen, muss diese nun in Fallstudien prototypenbasiert evaluiert werden. Hierzu muss zunächst der bereits entwickelte Prototyp für die Anforderungsanalyse erweitert werden, um wie in der Methode gefordert Abhängigkeiten von Anforderungen auch werkzeugseitig zu unterstützen. Aufgrund der dann in der Praxis gewonnenen Erkenntnisse kann die Werkzeugunterstützung verbessert und die Methode angepasst werden.

Neben der prototypischen Evaluierung kann es auch interessant sein, die Methode um weitere theoretische Konzepte zu ergänzen. So sind zwar domänenspezifische Methoden im RE schwierig zu gestalten, doch es sollte zumindest untersucht werden, ob domänenspezifische Instanzen durch den Einsatz von Ontologien und anderen semantikbasierten Technologien erzeugt und in der Praxis eingesetzt werden können.

Die Auswahl der Anforderungen könnte in der vorliegenden Methode zudem um Zeitaspekte erweitert werden, indem nicht nur der Nutzen und die Kosten, sondern auch die Entwicklungszeiten auf Ebene der Anforderungen einbezogen werden.

Des Weiteren kann eine Integration der Methode in das Produktlinien-Konzept [Böckle u.a. 2004] untersucht werden, um eine pro-aktive Wiederverwendung von Anforderungen zu ermöglichen. Hierbei ist insbesondere der Abgleich von Standardanforderungen mit wieder verwendbaren Komponenten ein spannendes Forschungsgebiet.

Um eine ganzheitliche kollaborative Methodik für das RE zu entwickeln, müssen in zukünftigen Arbeiten auch für die weiteren Phasen des RE Methoden für ein verteiltes Umfeld konzipiert werden. Eine solch ganzheitliche Methodik für das RE erlaubt dann eine stärkere Fokussierung von frühen Phasen der Softwareentwicklung, um so die zwischenbetriebliche Arbeitsteilung im Softwareentwicklungsprozess besser zu unterstützen, neue betriebliche Pro-

dukte und Prozesse schneller umzusetzen und die Qualität durch Einbeziehung mehrerer Akteure mit unterschiedlichen Fähigkeiten zu erhöhen. Zudem können durch eine stärkere Fokussierung des RE teure Folgefehler in späteren Phasen vermieden werden.

Die hier behandelte Problemstellung gilt aber nicht nur auf den RE-Prozess sondern erstreckt sich auf den gesamten Softwarelebenszyklus. Verteiltes Arbeiten – organisatorisch und/oder geographisch – spielt im weiteren Verlauf des Industrialisierungsprozesses in der Softwarebranche eine entscheidende Rolle. Es besteht daher Bedarf an einer durchgängigen methodischen und softwaretechnischen Unterstützung kollaborativer Softwareerstellung.

Literaturverzeichnis

[Alexander 1999] Alexander, Ian: Supporting a Co-operative Requirements Engineering Process. In: Proceedings of the 10th International Workshop on Database and Expert System Applications (DEXA'99), IEEE (1999), S. 340–344

[Beyer und Holtzblatt 1995] Beyer, Hugh R. ; Holtzblatt, Karen: Apprenticing With the Customer. In: Communications of the ACM 38 (1995), Nr. 5, S. 45–52

[Böckle u.a. 2004] Böckle, Günter ; Knauber, Peter ; Pohl, Klaus ; Schmid, Klaus : Software-Produktlinien: Methoden, Einführung und Praxis. dpunkt.verlag, 2004

[Boehm 2000] Boehm, Barry: Requirements that Handle IKIWISI, COTS, and Rapid Change. In: Computer 33 (2000), Nr. 7, S. 99–102

[Boehm und Bose 1994] Boehm, Barry W. ; Bose, Prasanta: A Collaborative Spiral Software Process Model Based on Theory W. In: Proceedings of the 3rd International Conference on the Software Process, Applying the Software Process, IEEE (1994)

[Boehm und Ross 1989] Boehm, Barry W. ; Ross, Rony: Theory W Software Project Management: Principles and Examples. In: IEEE Transaction of Software Engineering (1989), S. 902–916

[Briggs und Gruenbacher 2002] Briggs, Robert O. ; Gruenbacher, Paul: EasyWinWin: Managing Complexity in Requirements Negotiation with GSS. In: Proceedings of the 35th Hawaii International Conference on System Sciences (2002)

[Capgemini 2005] Capgemini: Studie IT-Trends 2005 – Paradigmenwechsel in Sicht. 2005. – Forschungsbericht. http://www.de.capgemini.com/servlet/PB/menu/1556859/ (30.08.2005)

[Cook und Churcher 2003] Cook, C. ; Churcher, N.: An Extensible Framework for Collaborative Software Engineering. In: Proceedings of the 10th Asia-Pacific Software Engineering Conference (APSEC'03), IEEE (2003), S. 290–301

[Damian 2001] Damian, Daniela: An empirical study of requirements engineering in distributed software projects: is distance negotiation more effective? In: Proceedings of the Asia Pacific Software Engineering Conference (APSEC) (2001)

[Damian u. a. 2003] Damian, Daniela E. ; Eberlein, Armin ; Shaw, Mildred L. ; Gaines, Brian R.: An exploratory study of facilitation in distributed requirements engineering. In: Requirements Engineering Journal: Special Issue on Selected Papers from RE'01 8 (2003), Nr. 1, S. 23–41

[Davis 1990] Davis, Alan M.: Software Requirements - Analysis and Specification. Prentice-Hall, 1990

[Dean u. a. 1998] Dean, Douglas L. ; Lee, James D. ; Pendergast, Mark O. ; Hickey, Ann M. ; Jay F. Nunamaker, Jr.: Enabling the Effective Involvement of Multiple Users: Methods and Tools for Collaborative Software Engineering. In: Journal of Management Information Systems 14 (1998), Nr. 3, S. 179–222

[ESI 1996] ESI: ESI Project No. 1100: ESPITI / European Software Institute. 1996. – Forschungsbericht. http://www.esi.es/en/Projects/VASIE/Reports/All/11000/ESPITI.doc (30.08.2005)

[French und Layzel 1998] French, Andy ; Layzel, Paul: A Study of Communication and Cooperation in Distributed Software Project Teams. In: IEEE Int. Conference on Software Maintenance, Bethesda (1998), S. 146–155

[Graham 1998] Graham, Ian: Requirements Engineering and Rapid Development. Addison-Wesley, 1998

[Gruenbacher 2000] Gruenbacher, Paul: Collaborative Requirements Negotiation with EasyWinWin. In: Proceedings of the 11th International Workshop on Database and Expert Systems Applications (DEXA'00) (2000)

[Gruenbacher 2002] Gruenbacher, Paul: EasyWinWin OnLine: Moderator's Guidebook, A Methodology for Negotiating Software Requirements. 2002. – URL http://sunset.usc.edu/research/WINWIN/EasyWinWin/ (30.08.2005)

[Gruenbacher und Braunsberger 2003] Gruenbacher, Paul ; Braunsberger, Patrick: Cooperative methods and tools for distributed software processes. Kap. Tool Support For Distributed Requirements Negotiation, S. 56–66, Franco Angeli, Milano, Italy, 2003

[Gruenbacher und Briggs 2001] Gruenbacher, Paul ; Briggs, Robert O.: Surfacing Tacit Knowledge in Requirements Negotiation: Experiences Using Easy Win Win. In: Proceedings

Hawaii International Conference on System Sciences, IEEE Computer Society (2001)

[Herbsleb u.a. 2001] Herbsleb, James D. ; Mockus, Audris ; Finholt, Thomas A. ; Grinter, Rebecca E.: An empirical study of global software development: distance and speed. In: Proceedings of the 23rd International Conference on Software Engineering, IEEE Computer Society (2001), S. 81–90

[Herlea und Greenberg 1998] Herlea, Daniela ; Greenberg, Saul: Using a Groupware Space for Distributed Requirements Engineering. In: Proceedings of the Seventh IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (1998), S. 57–62

[Highsmith 2000] Highsmith, James A.: Adaptive software development: a collaborative approach to managing complex systems. Dorset House Publ., New York, 2000

[Karlsson und Ryan 1997] Karlsson, Joachim ; Ryan, Kevin: A Cost-Value Approach for Prioritizing Requirements. In: IEEE Software 14 (1997), Nr. 5, S. 67–74

[Karlsson u. a. 1998] Karlsson, Joachim ; Wohlin, Claes ; Regnell, Björn: An evaluation of methods for prioritizing software requirements. In: Information and Software Technology 39 (1998), S. 939-947

[Karolak 1998] Karolak, Dale W.: Global software development: managing virtual teams and environments. Wiley-IEEE Computer Society Pr, 1998

[Leffingwell und Widrig 2000] Leffingwell, Dean ; Widrig, Don: Managing Software Requirements - A Unified Approach. Addison-Wesley, 2000

[Leuf und Cunningham 2001] Leuf, Bo ; Cunningham, Ward: The Wiki way: quick collaboration on the Web. Addison-Wesley, 2001

[Lloyd u.a. 2002] Lloyd, Wesley J. ; Rosson, Mary B. ; Arthur, James D.: Effictiveness of Elicitation Techniques in Distributed Requirements Engineering. In: Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'02) (2002)

[Mahmood 2003] Mahmood, M. A.: Advanced topics in end user computing. Idea Group Publishing, 2003

[Massey 2002] Massey, Bart: Where do open source requirements come from (and what should we do about it)? In: Proceedings of the 2nd ICSE Workshop On Open Source Software Engineering (2002)

[Massey 2003] Massey, Bart: Why OSS Folks Think SE Folks Are Clue-Impaired. In: Proceedings of the 3rd Workshop on Open Source Software Engineering, International Conference on Software Engineering (ICSE'03) (2003)

[Ocker u.a. 1995] Ocker, R. ; Hiltz, S.R. ; Turoff, M. ; Fjermestad, J.: Computer Support for Distributed Asynchronous Software Design Teams: Experimental Results on Creativity and

Quality. In: Proceedings of the 28th IEEE International Conference on System Sciences (1995), S. 4–13

[Partsch 1998] Partsch, Helmuth: Requirements-Engineering systematisch. Springer, 1998

[Pfahl 2001] Pfahl, Dietmar: An Integrated Approach to Simulated-Based Learning in Support of Strategic and Project Management in Software Organisation, Universität Kaiserslautern, Dissertation, 2001

[Ruhe u. a. 2002] Ruhe, Günther ; Eberlein, Armin ; Pfahl, Dietmar: Quantitative WinWin - A New Method for Decision Support in Requirements Negotiation. In: Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE 2002) (2002), S. 159-166

[Ruhe u. a. 2003] Ruhe, Günther ; Eberlein, Armin ; Pfahl, Dietmar: Trade-Off Analysis For Requirements Selection. In: International Journal of Software Engineering and Knowledge Engineering 13 (2003), Nr. 4, S. 345-366

[Saaty 1980] Saaty, Thomas L.: The Analytic Hierarchy Process. McGraw-Hill, 1980

[Scacchi 2002] Scacchi, Walt: Understanding the Requirements For Developing Open Source Software Systems. In: IEE Proceedings – Software, 149 (1) (2002), S. 24–39

[Scacchi u. a. 2004] Scacchi, Walt ; Jensen, Chris ; Noll, John ; Elliott, Margaret: Multimodal Modeling, Analysis and Validation of Open Source Software Requirements Processes. September 2004. – working paper, Institute for Software Research

[Seyff u. a. 2004] Seyff, Norbert ; Gruenbacher, Paul ; Maiden, Neil ; Tosar, Amit: Mobile Werkzeuge im Requirements Engineering. In: Softwaretechnik-Trends 24 (2004), 1

[Sommerville 2004] Sommerville, Ian: Software Engineering. 6. Auflage. AddisonWesley, 2004

[StandishGroup 1995] StandishGroup: CHAOS Report: Application Projects and Failures. 1995. – Forschungsbericht. http://www.standishgroup.com/ (30.08.2005)

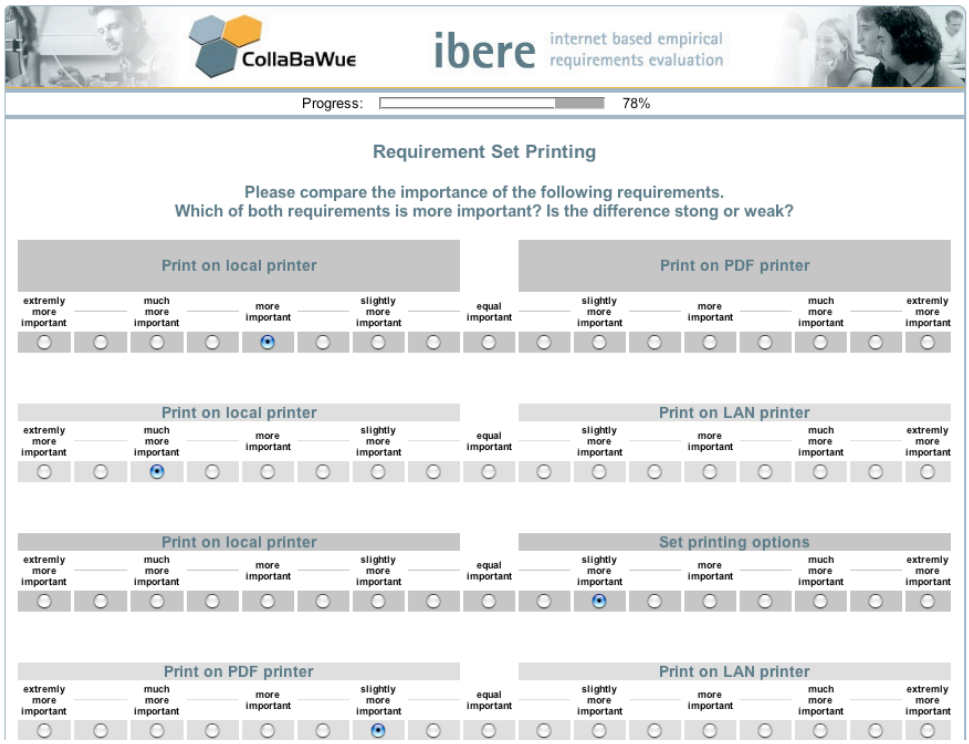
[StandishGroup 2001] StandishGroup: CHAOS Chronicles II. 2001. – Forschungsbericht. http://www.standishgroup.com/ (30.08.2005)

[Webster 2003] Webster, Melissa: Customer Needs and Strategies: An End-User View of the Collaborative Software Development Market / IDC. 2003. – Forschungsbericht. http://www.idc.com/ (30.08.2005)

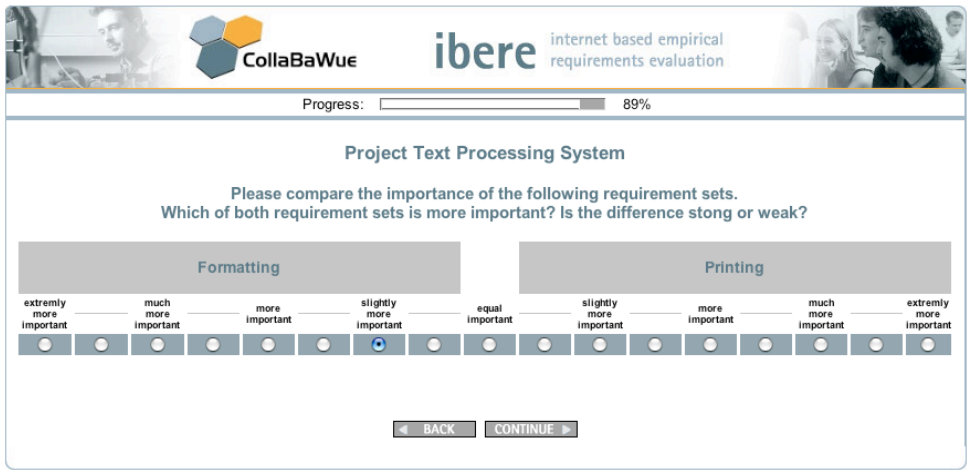
Anhang 1: Der Prototyp zur Anforderungsanalyse

Um die Schätzung der Kosten und des Nutzen zu unterstützen, wurde an der Universität Mannheim ein webbasierter Prototyp entwickelt, der die *Stakeholder* sicher durch den Prozess führt und die Nutzenwerte mit Hilfe des AHP berechnen und in einem Kosten-Nutzen-Diagramm visualisieren kann.

Es folgen zwei Screenshots, die die paarweise Bewertung des Nutzens von Anforderungen bzw. von Anforderungssets verdeutlichen (hierarchischer Einsatz des AHP).

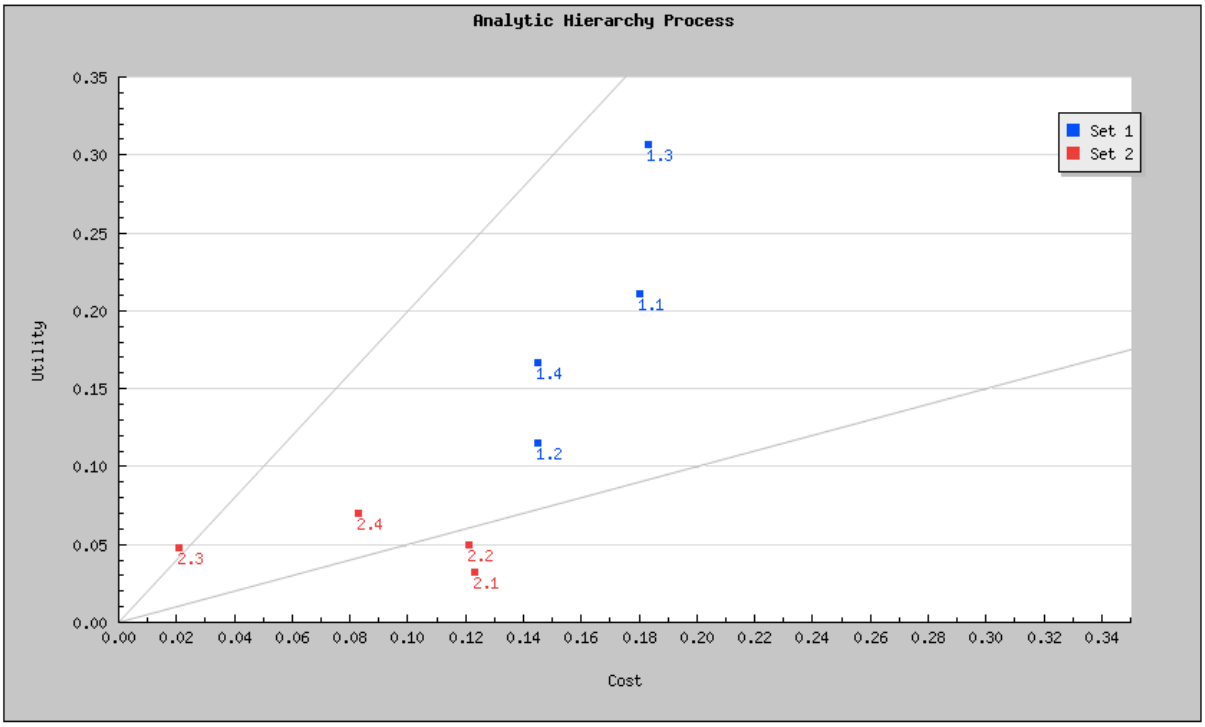


Screenshot 1: Bewertung des Nutzens der Anforderungen



Screenshot 2: Bewertung des Nutzens der Anforderungssets

Der dritte Screenshot zeigt die Visualisierungder Ergebnisse des AHP-Algorithmus. In diesem Beispiel sollte Anforderung 2.3 unbedingt implementiert werden, da sie oberhalb der Gerade y=2x liegt. Die Anforderungen 2.1 und 2.2 sollten daher nicht implementiert werden, da ihre Nutzen-Kosten-Verhältnisse ungünstig sind. Wie zu sehen ist, können mit dem Prototypen aktuell noch keine Abhängigkeiten von Anforderungen berücksichtigt werden. Dies soll mit der nächsten Version des Prototypen nachgeholt werden.



Screenshot 3: Visualisierung der AHP-Ergebnisse

Anhang 2: Die Methode in der Übersicht