Extreme Gebrauchstauglichkeit: Agile Prozess- und Softwareentwicklung mit den Mitteln des Usability Engineering

Hartmut Obendorf
C1 WorkPlace Solutions
Vogt-Kölln-Str. 30
22769 Hamburg
hartmut@obendorf.de

Ronald Naumann
Deutscher Ring
Ludwig-Erhard-Straße 22
20459 Hamburg
ronald.naumann@deutscherring.de

Abstract

Agile Entwicklungsmethoden versprechen durch die iterative Entwicklung von Anforderungen, Bedürfnisse von Kunden genauer treffen zu können als Ansätze des „klassischen“ Software-Engineering. Ausgehend von drei verschiedenen Projekten bei einem deutschen Versicherer berichten wir über unsere Erfahrungen mit der Ermittlung und Entwicklung von Anforderungen. Wir beschreiben das Fördern von Verbindlichkeit im erweiterten Entwicklungsteam und den Einsatz von Szenarien.

Keywords

Agiles Usability Engineering, Extreme Programming, Szenarien

1.0 Einleitung

Agile Entwicklungsmethoden versprechen nicht nur eine im Gegensatz zu klassischen Softwareprozessen erhöhte Produktivität, sondern auch die Ausrichtung von Softwaregestaltung an Wünschen und Nöten von Kunden und Endanwendern. Dabei sagen agile Methoden häufig nur wenig darüber aus, wie diese Anforderungen ausgehandelt und festgehalten werden oder wie daraus eine Vision für die Systemgestaltung abgeleitet werden kann. So bleibt den Anwendern agiler Entwicklungsmethoden viel Interpretationsspielraum.

Gerade für die Kommunikation mit Endanwendern und die Konsolidierung unterschiedlicher Nutzungsvorstellungen bietet das Usability Engineering (UE) eine Vielzahl von Methoden an. Das gewachsene organisatorische Umfeld in Versicherungen garantiert uns eine Vielzahl technischer Randbedingungen, so dass ein nicht mit der technischen Umsetzung rückgekoppeltes Design in den meisten Fällen nicht realisierbar ist – die Softwareentwicklung schafft so Rahmenbedingungen für das Usability Engineering. Damit wird die enge Integration von Usability- und Softwareentwicklung, neben der intensiven Kommunikation mit den Anwendern die Anpassung technischer Prototypen im Arbeitskontext notwendig.

1.1 Agile Methoden

Mit der Softwarekrise in den 1960er Jahren – die Kosten für Software überstiegen erstmals die von Hardware – wurde klar, dass die Entwicklung von Software ein sehr teures und in seiner Komplexität herausforderndes Unterfangen ist. Die Qualität von Software wurde wichtig (DIN ISO 55350) – zunächst vor allem die innere, die die Beschaffenheit der Software selbst betrifft, dann auch die äußere, die die Nutzung und den Einsatz bestimmt.

„Traditionelles“ Software-Engineering, wie es nicht zuletzt in großen Unternehmen bis heute umgesetzt wird, legt Wert auf einen geordneten Entwicklungsprozess, der formale Hilfsmittel einsetzt, um zu dokumentieren, wie Anforderungen in Software umgesetzt werden. Vorteile sind damit eine hohe Nachvollziehbarkeit und Planbarkeit des Entwicklungsprozesses. Es kann aber zu Geschwindigkeits- und Reibungsverlusten, zu einem Anspruch der Vollständigkeit (von Spezifikationen) kommen, der in der Praxis nicht eingehalten werden kann, und im ungünstigsten Fall wird das Einsatzumfeld statisch betrachtet – und so für Nutzer entwickelt, die es nach dem Ende der Entwicklung nicht mehr gibt, da sich die Anforderungen durch den Softwareeeinsatz geändert haben.

„Leichtgewichtige“ oder agile Methoden legen demgegenüber ein größeres Gewicht auf informelle Kommunikationswege (Beck 1999). Damit versprechen sie nicht nur eine schnellere Entwicklung, sondern vor allem schnelle erste Ergebnisse, die Scheitern oder Erfolg eines Projektes bereits frühzeitig erkennen lassen. Agile Methoden sind immer iterativ und setzen oft auf evolutionäres Prototyping, in dem Entwicklung und Einsatz verschränkt sind. Die praktische Relevanz agiler Methoden wird immer größer, neben Scrum (Schwaber 2001) und FDD (de Luca 2008) hat vor allem XP (Beck 1999, 2004) eine wichtige Rolle in der Softwareentwicklung erreicht.

1.2 Extreme Gebrauchstauglichkeit – UE-Techniken und agile Prozesse

Die Kombination von Usability Engineering (UE) und agiler Softwareentwicklung liegt aufgrund der zunehmenden Bedeutung der agilen Methoden nahe und wird intensiv diskutiert. UE wird dabei oft als vorbereitende Phase für die Entwicklung gesehen (z.B. (Memmel et al. 2006)) – dies birgt jedoch die Risiken des Zeitverlustes für die Ausführung des UE und des Ausbremsens der agilen Softwareentwicklung (Obendorf & Finck 2007). Die Integration von UE-Techniken in agile Methoden ist aufgrund der Phasenbezogenheit des UE nicht trivial (Nebe et al. 2007).

In diesem Beitrag leiten wir aus Schwierigkeiten bei der Umsetzung unserer Implementation von XP und UE unsere Forderung nach einer höheren Verbindlichkeit der Anforderungen ab und beschreiben den Einsatz von Szenarien, um dem zu begegnen. Als Basis dient ein Prozessmuster, welches auf Szenariotechniken aufsetzt, um die „Stories“ von Extreme Programming (XP) im Kontext zu verankern und die „Konversation mit dem Kunden“ zu moderieren (Obendorf et al. 2005, Obendorf & Finck 2008).

2.0 Fallbeispiele für agile Entwicklung beim Deutschen Ring

Der Deutsche Ring ist ein traditionelles Versicherungsunternehmen mit Stammsitz in Hamburg. Zur Unternehmensgruppe gehören neben der Lebens-, Kranken-, Unfall- und Sachversicherung die MoneyMaXX AG, die Deutscher Ring Bausparkasse, die Fondsgesellschaft Financial Services, die Maklermanagement AG, der Deutsche Pensionsring sowie einige große Vertriebsgesellschaften. 2008 belegte der Deutsche Ring Krankenversicherungsverein a.G. beim Wettbewerb "Deutschlands kundenorientiertester Dienstleister" den 3. Platz und ist damit Deutschlands kundenorientiertester Versicherer (Handelsblatt vom 28.5.2008).

In der Entwicklung webbasierter Anwendungen wird vom Deutschen Ring seit zwei Jahren intensiv XP als agile Vorgehensmethode eingesetzt, wobei wie in jedem traditionsreichen Unternehmen sehr verschiedene Entwicklungsparadigmen nebeneinander stehen, und so z.B. auch weiterhin in vielen Bereichen Großrechnertechnik und klassische Softwareentwicklungsmodelle betrieben werden. Dies bringt mehrere Besonderheiten mit sich, die eine Anpassung des XP-Prozesses an die Organisationskultur erforderlich machen: Projekte werden aufgrund sehr global formulierter Anforderungen geplant und beauftragt, oftmals sind viele verschiedene Abteilungen an einem Projekt beteiligt und auch die Kundenrolle ist zumeist auf verschiedene Organisationseinheiten verteilt.

Die hier beschriebenen Erfahrungen stammen aus drei verschiedenen Projekten, die kurz skizziert werden sollen: Das Makler-Auskunft- und Controllingsystem (**MACS**) ist die zentrale Informationsplattform für die Betreuer und Servicemitarbeiter des Maklervertriebs des Deutschen Rings. Stamm-, Kontakt- und Provisionsdaten der Makler werden aus einer Vielzahl anderer Systeme und Datenbanken gesammelt und in MACS angezeigt. Die verschiedenen Bearbeitungsstände der vom Makler eingereichten Versicherungsanträge und die Aufgaben des Maklerservice bezogen auf die einzelnen Anträge werden in MACS verwaltet.

Die elektronische Kundenakte (**eAkte**) ist als Hauptarbeitsmittel für die Kundenbetreuung im Servicemanagement angelegt. Es handelt sich dabei um ein Informationssystem, in dem Daten aller Verträge, an denen ein Kunde als Versicherungsnehmer oder versicherte Person beteiligt ist, kundenbezogen und spartenübergreifend in ihrem aktuellen Zustand dargestellt werden können.

Das **Maklerportal** ist ein Informationssystem, welches auf die Informationen von MACS und der elektronischen Kundenakte zugreift und die Bestandsdaten einzelner Makler für diese bereitstellt. Damit können Verarbeitungszustand wie Vertragsdetails aller beim Deutschen Ring eingereichten Anträge und Verträge vom Makler angefordert werden. Im Maklerportal ist außerdem noch der Zugriff auf Werbemittel und Angebotsrechner möglich, um den Makler im Vertrieb zu unterstützen.

Bei MACS wie der eAkte handelt es sich um Komponenten der hausinternen Informationsplattform iKontakt, das Maklerportal ist eine moderne Eigenentwicklung. Alle drei Anwendungen basieren trotz sehr unterschiedlicher Frontend-Technologien auf einer gemeinsamen Serviceschicht, die den Zugriff auf verschiedene Datenbanken und die beim Deutschen Ring im Einsatz befindlichen Hostrechnersysteme kapselt.

3.0 Verbindlichkeit

Beim Deutschen Ring wurde und wird ein großer Teil der Softwareentwicklung nach klassischen Methoden durchgeführt. Die beschriebenen Projekte waren die ersten, die agil ausgelegt waren und sich im Vorgehen an einer Kombination von XP mit Szenariotechniken orientierten (vgl. Obendorf & Finck 2008). Das iterative Vorgehen mit vierwöchigen Iterationsmeetings mit Projektleiter, Auftraggeber, Kunden und Entwickler bedeutete eine Umstellung für alle Beteiligten. Nach jeder Iteration wurde eine neue Softwareversion zum Testen zur Verfügung gestellt und nach Behebung der aufgetretenen Fehler produktiv eingesetzt. In der konkreten Umsetzung von XP setzen wir den *XPlan-*

ner[1] (www.xplanner.org) für die Verwaltung der Stories und die daraus abgeleiteten Tasks ein, in einem *Wiki* können Kunden, Nutzer und Entwickler gleichberechtigt aktuelle und historische Stände der Stories einsehen und bearbeiten.

Alle beschriebenen Projekte wurden hausintern mit externer Unterstützung entwickelt, weshalb die Entwicklungskapazität von Beginn an festgelegt war. Damit stellt sich jeweils nicht die Frage nach der Umsetzung aller erfassten Anforderungen, sondern nur danach, welche Anforderungen im Projektrahmen umgesetzt werden können. Die Herausforderung für unseren Kunden besteht also darin, die einzelnen Stories zur Umsetzung im Rahmen der Iterationen "richtig" zu priorisieren, damit am Ende die Funktionalität umgesetzt wird, die wirklich am dringendsten benötigt wird. Für unsere Kunden macht dieses für sie neue Vorgehen zunächst Vieles einfacher. Sie können auch noch während des Projektes Anforderungen beschreiben und zu jeder Iteration in Abhängigkeit vom erreichten Entwicklungsstand neu priorisieren. Die Folgen von Entscheidungen werden durch die unmittelbare Rückmeldung sichtbar und Korrekturen erfolgen zu Beginn einer jeden Iteration fast zwangsläufig.
Der Projektstand insgesamt ist für alle Beteiligten nachvollziehbar, Zeitüberschreitungen sind früher erkennbar. Geeignete Gegenmaßnahmen (Verschiebung einzelner Stories in spätere Releases, Anforderung zusätzlicher Ressourcen) können zielgenauer erfolgen.
Eines der anfänglichen Probleme innerhalb des Projektes MACS bezogen auf die Vorgehensweise war das heterogene Auftreten der Kunden. So wurden in den ersten Wochen von unseren Kunden besondere Suchfunktionen und die sehr differenzierte Darstellung von Kommunikationsdaten der Makler sehr hoch priorisiert. Im Laufe des Projektes hat sich herausgestellt, dass diese Funktionen nur von einem engen Nutzerkreis und sehr selten verwandt werden. Gemeinsam mit dem Kunden konnten wir das Verhältnis der Nutzer ausgleichen und so die Prioritäten den Bedeutungen der einzelnen Anforderungen annähern.

3.1 Verbindlichkeit: Vollständigkeit

Die fortlaufende Beschreibung und Priorisierung von Anforderungen hat aber auch Nachteile: Unser Kunde begann, gerade zu wichtigen und damit scheinbar klaren Anforderungen eine Konsumentenhaltung einzunehmen und die Beschreibung der Anforderung nicht ausreichend auszuführen: Unser Kunde hat z.B. anstelle einer vollständigen Beschreibung einer Funktion lediglich einige wenige Ideen beschrieben und anhand der erfolgten Umsetzung im Anschluss seine Beschreibung weiter komplettiert. Notwendiger Klärungsbedarf ging oft zu Lasten der Entwickler, da die Vollständigkeit der Anforderungen durch deren Nachfragen und spätere Beschreibung sichergestellt werden musste.
Vorgehenskonforme, später zu beschreibende Anforderungen ließen sich in einigen Fällen nicht mehr von Change Requests (also nachträglichen Anforderungen) abgrenzen. Damit nahm die Umsetzung einiger Funktionen ein Vielfaches der geplanten Zeit in Anspruch. Hinzu kamen weitere Abstimmungs- und Testaufwände.
Eine wichtige Schlussfolgerung für die weitere Arbeit mit leichtgewichtigen Methoden beim Deutschen Ring ist daher, dass Anforderungen zwar in anderer Form (Story) und in einem anderen Kontext (kleinteilig und wo möglich auch unabhängig voneinander) beschrieben werden sowie zu einem anderen Zeitpunkt (z.T. nach Beginn des Projektes).
Jede einzelne Story sollte, bevor sie zur Umsetzung priorisiert wird, so vollständig wie möglich und unter Berücksichtigung aller denkbaren Ausnahmen beschrieben worden sein. In einem traditionellen Entwicklungsprozess entspricht dies einem Quality Gate; in agilen Prozessen ist die größte Herausforderung die Identifikation der Kandidaten für neue Stories, denn für diese muss vor der Iterationsplanung mit dem Kunden ein Klärungsprozess eingeleitet werden.

3.2 Erweitertes Entwicklungsteam

Nach unseren Erfahrungen – bestimmt durch die hohe Komplexität des Projektumfeldes und vielen beteiligten Stakeholdern – ist die Rollenverteilung Entwicklerteam vs. Kunde, wie sie in XP vorgesehen ist, nicht ausreichend. Der „Kunde vor Ort“, wie er noch in der ersten Version von XP beschrieben wurde, ist als ein Hauptdefizit von XP ausgemacht worden (z.B. McBreen 2002) und drückt eine unrealistische Vorstellung aus. Stattdessen schlagen wir vor, ein je nach Aufgabe variabel aufgestelltes *erweitertes Entwicklungsteam* zu definieren, in dem Kunde und Benutzer wichtige Rollen ausfüllen: Der *Auftraggeber* kommt bei uns aus dem gleichen Bereich wie die Mehrzahl der Benutzer, ist aber entfernt von den eigentlichen operativen Anforderungen.

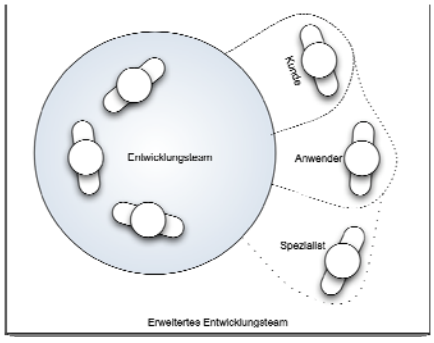

Abb. 1: Erweitertes Entwicklungsteam

[1] Auch die Zuweisung der Entwickler zu einzelnen Tasks, Aufwandsschätzungen und Verbuchung angefallender Aufwände werden hier vorgenommen.

Der *Nutzer* erfährt die Fachlichkeit im Rahmen seiner täglichen Arbeit sehr genau, kann aber wenig zur zukünftigen Ausrichtung des Bereichs sagen und besitzt nicht die Entscheidungskompetenz des Auftraggebers.

Der *Spezialist* kennt die Fachlichkeit sehr genau, inkl. aller Ausnahmen, rechtlicher Bedeutung etc. Er ist aber nicht Zielgruppe der Software und wird nicht oder selten damit arbeiten.

Ziel des erweiterten Entwicklungsteams ist es, für die Beschreibung von Anforderungen eine hohe Verbindlichkeit herzustellen, so dass der Kunde, Anwender und Spezialist gleichermaßen das Gefühl bekommen, dass ihre Anforderungen berücksichtigt werden.

Mitglieder in diesem Team sind neben den jeweiligen Entwicklern Vertreter der drei skizzierten Gruppen. Sie alle haben gleichberechtigt Zugriff auf die Stories.

Im Vorfeld stellt der Nutzer eine Reihe von Anforderungen aus Sicht seiner täglichen Arbeit auf. Er benennt diese Abläufe und Funktionalitäten. Der Auftraggeber sammelt, priorisiert und fordert von den Entwicklern eine erste grobe Aufwandsschätzung für die aus seiner Sicht wichtigen Punkte an.

Zur inhaltlichen Abstimmung einer konkreten Anforderung treffen sich Auftraggeber, Nutzer, Spezialist und Entwickler und beginnen, die Story gemeinsam und detailliert zu beschreiben. Dabei wird darauf geachtet, dass die Beschreibungen entweder direkt im Wiki vorgenommen werden oder unmittelbar im Anschluss dort erfolgen. Anhand dieser Beschreibungen wird ein Autor-Kritiker Zyklus in Gang gesetzt, mit dem Ziel eine möglichst vollständige Beschreibung der Story zu bekommen.

Das Entwicklungsteam konkretisiert die Aufwände für die Umsetzung anhand der detaillierteren Beschreibung und schafft somit die Voraussetzung, die jeweilige Story in der kommenden Iterationsplanung zu berücksichtigen.

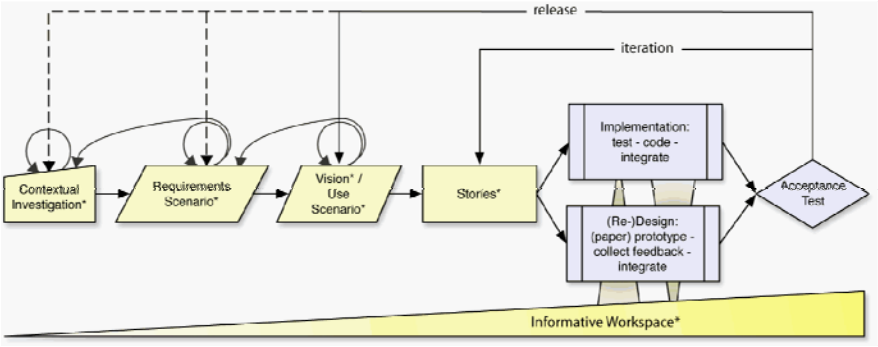

Abb. 2: Das implementierte Prozessmuster (vgl. Obendorf & Finck 2008)

4.0 Anforderungserhebung: Szenarien und Stories

Mittel der Wahl für die Dokumentation von Anforderungen sind in XP *Stories*, der Idee nach kurze Geschichten, die eine benötigte Funktionalität beschreiben (z.B. „ich melde mich an). Stories sind bewusst unvollständig gehalten, ihre genaue Ausgestaltung soll erst im Laufe der Entwicklung und im Dialog von Kunde / Anwender und Entwicklerteam verhandelt werden.

Dabei gibt es zwei prinzipielle Schwierigkeiten: zum einen ist die Trennung der Rollen Kunde / Anwender in XP nicht konsequent: Da die Verantwortung für die Anforderungen beim Kunden liegt, besteht die Gefahr, an den Anwendern vorbei zu entwickeln. Auch ist der Gesamtzusammenhang einer Story oft unklar, ihr fehlt die Verbindung zur Gesamtvision.

4.1 Entwicklung der Anforderungen mit Stories

Daher beschreiben wir hier, wie wir die gemeinsame Erstellung von Stories und Szenarien einsetzen, um Anforderungen zu entwickeln und im Nutzungskontext zu verorten.

XP schlägt vor, die wichtigsten Funktionen eines Systems zu realisieren, um dann Funktionalität sinnvoll zu ergänzen. Dies birgt aber die Gefahr, keine ganzheitliche Vorstellung zu entwickeln und so nur lokal sinnvolle Ergebnisse zu erreichen (Obendorf & Finck 2008).

Eine einfache Übertragung von UE-Techniken würde vor dem Beginn eines XP-Prozesses versuchen, Anforderungen möglichst vollständig zu erfassen. Will man agil bleiben – und sollen Nutzer und Entwickler im erweiterten Entwicklungsteam mitwirken, kann aber dabei nicht einmal annähernde Vollständigkeit erreicht werden, sondern Kontextanalyse und das Design müssen ebenso iterativ und schrittweise erfolgen. Damit wird es notwendig, neben den *Rohstories*, die als „promise for conversation“ (Beck 1999) dienen und eine erste Planung ermöglichen, *konsolidierte Stories* als Grundlage für die Entwicklung und ggf. auch *Szenarien* für die Kontextualisierung der Stories einzusetzen.

Beim Deutschen Ring gibt es einige zusätzliche Rahmenbedingungen, die die Nutzung von Stories bestimmen: Dadurch, dass nicht alle beteiligten Abteilungen ein agiles Prozessmodell verfolgen, setzte der XP-Prozess immer auf vorangehenden Planungsschritten auf. So war die Grobplanung der Projekte z.T. bei Start schon abgeschlossen; es lag zwar nie ein vollständiges Anforde-

rungsdokument vor, für das MACS-Projekt existierte aber z.B. eine Anforderungsdatenbank, für die sehr grob Aufwände geschätzt worden waren. Für das Projekt eAkte gab es dagegen nur detaillierte und oft fehlerhafte inhaltliche Vorgaben, die Spezifikation der Stories wurde vor Projektbeginn durch die Kunden durchgeführt.

Wir haben ein mehrstufiges Verfahren zum Erheben und Klären der Anforderungen eingesetzt: von *rohen Geschichten*, die oftmals nur stichpunktartig die Vorstellungen der Kundenvertreter beschreiben (und damit bereits die im Vorfeld erhobenen Benutzerwünsche interpretieren und filtern), versuchen wir iterativ und gemeinsam mit den zukünftigen Anwendern zu *konsolidierten Stories* zu gelangen, die dann nicht nur die Anforderungen beschreiben, sondern auch Grundlage für die Abnahme und Dokumentation des entwickelten Systems sind.

4.2 Arbeitskontext und Gesamtzusammenhang: Stories und Szenarien

Eine wichtige Eigenart unseres XP-Prozesses ist die Entwicklung von Stories aus einem Rohzustand hin zu einer brauchbaren Umsetzungsvorgabe und -dokumentation. Eine zentrale Rolle kommt so Szenarien zu, die als kontextuelle Beschreibung der Nutzung von zukünftiger Funktionalität nicht nur für die Umsetzung wichtige Details beinhalten, sondern vor allem Nutzungsumfeld und Hintergründe der Nutzung transportieren. So geben sie einzelnen Funktionalitäten einen Rahmen, illustrieren gegenseitige Abhängigkeiten und erhöhen so die Planbarkeit. Ein Szenario kann dabei mit mehreren Stories verbunden sein und beschreibt Tätigkeiten, die benötigte Funktionalitäten begründen und bestimmen.

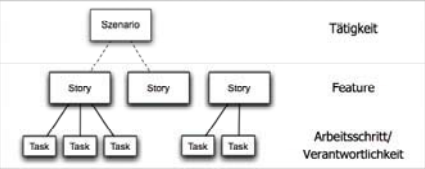

Abb. 3: Bezug von Szenarien und Stories.

An anderer Stelle setzen wir Szenarien ein, um gezielt Defizite, die in Rohstories offensichtlich werden, zu beschreiben. Im Rahmen des Projekts eAkte nutzt ein Sachbearbeiter z.B. bis zu fünf verschiedene Auskunftssysteme, in den ursprünglichen Stories waren aber nur die Anforderungen für das neue System beschrieben. In diesen Fällen dienen Szenarien vor allem dazu, den Workflow zu beschreiben und so die Rationalität von Designentscheidungen offen zu legen.

Nicht immer ist der Einsatz von Szenarien erforderlich – sind etwa nur wenige Stakeholder an einer Entscheidung beteiligt und hat neue Funktionalität nur sehr lokale Auswirkungen, belassen wir die mögliche Ausdifferenzierung in der direkten, mündlichen Aushandlung mit dem Benutzer.

5.0 Fazit

Der Einsatz des vorgestellten Vorgehens wird von allen Beteiligten – auch den Kunden – sehr positiv eingeschätzt. Dennoch sind wir auf Probleme gestoßen, für die wir teilweise Gegenmaßnahmen gefunden haben.

Dazu gehört die Gefahr, dass die stetige Entwicklung zu einem Eindruck der Unverbindlichkeit führt. Hier haben wir uns zum einen entschieden, Kunden und Benutzer in einem *erweiterten Entwicklungsteam* einzubinden, zum anderen Qualitätsanforderungen für die einzelnen Dokumente aufzustellen und im Entwicklungsprozess Zeitpunkte festzulegen, zu denen eine Mindestqualität erreicht sein muss (Quality Gates).

Szenarien ergänzen für uns den XP-Prozess, einerseits, indem sie Stories verorten und die Planbarkeit erhöhen, andererseits, indem sie den gemeinsamen Aushandlungsprozess für Anforderungen unterstützen und die Qualität der Stories nachhaltig verbessern.

5.1 Literaturverzeichnis

Beck, K. & Andres, C. (2004). *Extreme Programming Explained : Embrace Change (2nd Ed.).* Addison-Wesley Professional.

Beck, K. (1999). *Extreme Programming Explained: Embrace Change*. Addison Wesley.

Beck, K., Beedle et al. (2001). *Agile Manifesto*. http://www.agilemanifesto.org.

Cohn, M. (2004). *User Stories Applied: For Agile Software Development.* Addison Wesley.

de Luca, J. (2002). *FDD processes.* http://www.nebulon.com/articles/fdd/.

Gundelsweiler, F., Memmel, T., & Reiterer, H. (2004). *Agile Usability Engineering*. Mensch & Computer 2004, München, 33-42.

McBreen, P. (2002). *Questioning Extreme Programming.* Pearson Education.

Nebe, K., Düchting, M. & Zimmermann, D.: *Integration von User Centred Design Aktivitäten in Agile Softwareentwicklung*. Usability Professionals 2007.

Obendorf, H. & Finck, M. (2008). *Scenario-Based Usability Engineering Techniques in Agile Development Processes*. CHI '08: ACM Conference on Human Factors in Computing Systems, Florence, 2159-2166.

Obendorf, H.; Finck, M.; (2007). *Szenariotechniken & Agile Softwareentwicklung.* Mensch & Computer 2007 - Interaktion im Plural. München, 19-28.

Obendorf, H., Finck, M., & Schmolitzky, A. (2005). Teaching balance and respect. *Interactions*, *12*(5), 36-37.

Schwaber, K. & Beedle, M. (2001). *Agile Software Development with SCRUM.* Prentice Hall.

Eloquenz in der Interaktion