Integration von User Centered Design in agilen Entwicklungsprozessen

Markus Düchting
Siemens IT Solutions
C-LAB
Fürstenallee 11
33102 Paderborn
markus.duechting@c-lab.de

Petra Kowallik
Open Text Corp.
Product Design
Werner-von-Siemens Ring 20
85630 Grasbrunn b. München
petra.kowallik@opentext.com

Karsten Nebe
Universität Paderborn
C-LAB
Fürstenallee 11
33102 Paderborn
karsten.nebe@c-lab.de

Abstract

Der vorliegende Beitrag behandelt die Integration von User Centered Design und agiler Softwareentwicklung. Es wurden existierende Ansätze betrachtet und gemeinsame Kernaspekte dargelegt. Diese werden im Bezug zur Praxis an einem konkreten Anwendungsfall verdeutlicht.

Im Praxisteil wird Open Text's Agile Design-Methode beschrieben, bei der der Nutzungskontext in frühen Research Phasen evaluiert und mit Hilfe der Persona Methode dargestellt wird. Der iterative Entwicklungsprozess bildet den Rahmen für ein agiles Design, das eine hervorragende User Experience liefert.

Keywords

Agile Software Engineering, Agile Design, User Centered Design

1.0 Einleitung

Die agile Softwareentwicklung gewinnt in den letzten Jahren immer mehr Aufmerksamkeit, sowohl seitens der Softwareindustrie als auch der Wissenschaft. Dies zeigen beispielsweise die zahlreichen Konferenzen bzw. die öffentlichen Bekundungen vieler Unternehmen agile Entwicklungsprozesse einzusetzen.

Die wachsende Verbreitung von agilen Vorgehensweisen in der Softwareentwicklung zieht auch die Aufmerksamkeit der HCI Community auf sich. So resultiert daraus die Frage, ob Softwaresysteme die agil erstellt werden, gebrauchstauglich sind? Vielleicht bietet das Agile sogar dafür eine besondere Chance? Könnten sich Probleme, die in der klassischen Softwareentwicklung existieren, vielleicht im Agilen auflösen? Rigide definierte, lineare Projektpläne, monolithische Phasen, umfangreiche Artefakte / Dokumentation werden z. B. gegenüber der Interaktion mit den Kunden in den Hintergrund gestellt. Viele dieser Aspekte werden auch bereits durch Vorgehensweisen des UCD adressiert.

In der Literatur existiert zudem eine Vielzahl von Integrationsansätzen, die UCD und agile Vorgehensweisen kombinieren. Es besteht offensichtlich ein Handlungsbedarf seitens der UCD Community auf diese Entwicklung angemessen zu reagieren und die Chance zu nutzen, qualitativ hochwertige und gebrauchstaugliche Produkte agil zu entwickeln.

Der Fragestellung, inwieweit agile Modelle bereits Aspekte des Usability Engineering berücksichtigen, haben die Autoren bereits im Rahmen einer vorhergehenden Betrachtung diskutiert (Nebe et al 2007). Daraus entstand die Motivation die existierenden Integrationsansätze genauer zu betrachten und einander gegenüber zu stellen.

2.0 Integrationsaspekte

Im Folgenden werden Kernaspekte aufgeführt, welche auf die wesentlichen Gemeinsamkeiten der unterschiedlichen Ansätze abzielen. Dazu werden zunächst die identifizierten Kernaspekte allgemein beschrieben, schließlich einige spezifische Besonderheiten bei der Integration vorgestellt und anschließend mit Beispielen aus dem Vergleich verdeutlicht. Die daraus resultierenden Forderungen werden im Anschluss gesammelt vorgestellt und der Bezug zur Praxis an einem konkreten Anwendungsfall verdeutlicht.

2.1 Analyse

In der Softwareentwicklung werden in einer Analysephase Informationen über den zukünftigen Nutzer gesammelt und zusammengefasst. Dazu zählen die Aufgaben und Ziele der Nutzer, die Arbeitsabläufe und der Nutzungskontext, einschließlich eventueller Umwelteinflüsse oder technischer Rahmenbedingungen.

Im User Centered Design wird es als unerlässlich angesehen, die zukünftigen Nutzer eines Systems so gut wie möglich zu kennen, bevor mit der Entwicklung von Designs und der Implementierung begonnen wird. Unterschied zum Agilen ist, dass i.d.R. keine explizite, vorgelagerte Analysephase durchgeführt wird. Dies wird versucht, durch die permanente Anwesenheit eines Customer Stakeholder, auszugleichen. Bei einigen der betrachteten Integrationsansätze

Brau, H., Diefenbach, S., Hassenzahl, M., Koller, F., Peissner, M. & Röse, K., (Hrsg.): Usability Professionals 2008

wurde die Analyse des Nutzungskontexts auch in Kombination mit dem Agilen beibehalten. Diese Analysen sind jedoch nicht so umfangreich, wie sie in klassischen UCD Prozessen vorgesehen werden. Vielmehr wird die Idee der agilen Entwicklung aufgegriffen, nur so viel Wissen zu sammeln, wie für die nächste Iteration entscheidend ist. Dabei entstehen schlankere Dokumente und das Wissen wird oft persönlich weitergegeben. Unterschiede in den Ansätzen existieren darin, wie lang diese Phasen sind, die Art der Dokumentation, wer die Aktivitäten durchführt, ob sie parallel zur Entwicklung stattfinden oder in einem zeitlich versetzten Zyklus durchgeführt werden (Chamberlain et al 2006; Detweiler 2007; Sy 2007).

Im Ansatz von Beyer et al. (2004) wird beispielsweise eine Kontextanalyse (Contextual Inquiry) mit mindestens drei Personen in unterschiedlichen Nutzerrollen durchgeführt, die ca. eine Woche dauern soll. Um den Aufwand für den Wissenstransfer gering zu halten, wird die Analyse von einem interdisziplinären Team aus UI Designern, Entwicklern, Marketing Repräsentanten, etc. durchgeführt.

Generell ist das Konzept der Analyse, wie es im UCD verstanden wird, nicht konsistent mit dem iterativen Vorgehen bei einer agilen Vorgehensweise. Auf der einen Seite wird im UCD ein iteratives Vorgehen propagiert und auf der anderen Seite eine monolithische und lineare Phase am Anfang des Prozesses eingefordert. Für einen schnellen Entwicklungsprozess wäre es von Vorteil, wenn die Analyse aus dem Produktentwicklungsprozess heraus gelöst wird und bereits vorher von Experten durchgeführt würde. Wichtig aus Sicht des UCD ist es, dass dieses Wissen im Vorfeld vorhanden ist und nicht, ob dies im selben Entwicklungsprozess stattfindet [Norman 2006].

Haikara (2007) beschreibt in einer Fallstudie mehrere agile Projekte, in denen die Entwicklung durch Persona-Beschreibungen unterstützt wurde. Dabei zeigte sich, dass die zeitintensive Erstellung von umfangreichen Personas die agile Entwicklung aufzuhalten drohte.

Dies kann als Indiz dafür gewertet werden, dass es Sinn macht, die Analyse vom eigentlichen Entwicklungsprozess zu entkoppeln. um diesen nicht auszubremsen.

2.2 Modellierung

Bei der Modellierung geht es in erster Linie darum, Rollen, Aufgaben, Ziele und Anforderungen im (größeren) Zusammenhang darzustellen, um von der Abbildung eines IST-Zustandes zu einem verbesserten SOLL-Zustand zu gelangen.

Die kleinschrittige, inkrementelle Entwicklung im Agilen erschwert eine ganzheitliche Modellierung von Interaktion und Arbeitsabläufen. Es besteht die Gefahr, dass Anforderungen bzgl. des Workflows der Nutzer nicht in ihrem Gesamtkontext gesehen werden, wenn sie lediglich in Form von User Stories oder Backlog Einträgen festgehalten werden. In der Umsetzung muss daher sichergestellt sein, dass nicht nur einzelne Arbeitsabläufe der Nutzer erfasst und implementiert werden, sondern auch dass die Beziehungen zwischen den Arbeitsabläufen der Nutzer im konkreten Handlungskontext berücksichtigt werden.

In den untersuchten Integrationsansätzen wurden dafür unterschiedliche Artefakte, wie Use Cases, Scenarios, Task Models und auch Personas verwendet (Chamberlain et al 2006 ;Patton 2002; Wolkerstorfer et al 2008).

Constantine & Lockwood (2003) schlagen beispielsweise vor, die Tasks der Nutzer in Form von Use Case Szenarien auf Index-Karten zu formulieren, diese zu gruppieren und zu priorisieren. Wolkerstorfer et al. (2008) beschreibt hingegen in einer Fallstudie, wie die Entwicklung durch Persona-Beschreibungen unterstützt wurden. Die (Extreme)-Personas in diesem Beispiel zeichnen sich durch ihre inkrementelle Entwicklung und Refaktorisierung aus. Daher auch die Bezeichnung *Extreme* Persona.

2.3 Testing und Prototyping

Das kontinuierliche und detaillierte Testen von Teil-Lösungen, ist im Agilen mit dem Ziel verankert, weitestgehend fehlerfreie Software sicherstellen zu können. Dies wird auf technischer Seite durch automatisierte Unit-Tests realisiert und aus Kunden/Nutzer-Sicht durch sog. Acceptance Tests, bei denen ein Softwaresystem gegen die im Vorfeld definierten Anforderungen evaluiert wird.

Aus Usability Sicht sind solche automatisierten Test nur schwer möglich und beschränken sich z. B. auf die Prüfung der Konformität des UI (z. B. der richtigen Beschriftung von Interfaceelementen). Das automatisierte Testen komplexer Workflows ist jedoch kaum möglich.

In agilen Modellen ist i.d.R. nicht verankert während einer Prozessiteration Zwischenergebnisse, die noch nicht in Code umgesetzt wurden, mit Nutzern oder anderen Stakeholdern zu überprüfen. Die Betrachtung der Ansätze hat gezeigt, dass gerade für die Entwicklung einer gebrauchstauglichen Nutzerschnittstelle die kontinuierliche Evaluation mit Nutzern maßgeblich ist und Teil des Modells sein sollte. Eine häufig angewandte Methode scheint das Paper Prototyping. Der Vorteil ist ein schnelles Feedback ohne den Ballast eines formalen Tests.

Im Rapid Contextual Design Ansatz (Beyer et al. 2004) werden Paper Mock-Ups verwendet, um erste Designentwürfe mit Nutzern zu überprüfen, bevor diese in Code umgesetzt werden. Patton (2008) sieht einen deutlichen Mehrwert in unformalen Tests mit Papierprototypen und beschreibt, wie ein solcher Validierungs-Prozess in einen Design-Prozess übergeht. Ein Papierprototyp bietet ihm die Möglichkeit, auf Feedback direkt einzugehen, in dem unmittelbar Veränderungen vorgenommen werden können, um diese wiederum zu validieren. Kane (2003) befürwortet ein Szenario-basiertes Prototyping, auf Grund der geringen Kosten und weil es nicht von lauffähiger Software abhängig ist.

2.4 Verortung von Aktivitäten (Parallelität, Nebenläufigkeit)

In der traditionellen Softwareentwicklung werden Aktivitäten oft schritt- oder phasenweise, meist auch sequentiell, abgearbeitet. Der Entwurf startet meist nach der Analyse, und die Implementierung nach dem Entwurf. Abschließend wird die entwickelte Teil-Lösung getestet. Dies führt dann zu einer Release-Version, die an Kunden ausgeliefert wird. In der agilen inkrementellen Entwicklung wird das Ergebnis einer jeden Prozessiteration als potentielles Mini-Release verstanden. Dabei werden vergleichbare Aktivitäten wie in der traditionellen Softwareentwicklung durchgeführt, aber kleinschrittiger; die Lösungserarbeitung erfolgt sukzessive. Die Software, die am Ende einer Iteration vorliegt, soll nur aus vollständig getesteten und lauffähigen Features bestehen.

In einer Fallstudie beschreibt Sy (2007) den agilen Design und Entwicklungsprozess bei der Firma Autodesk Inc. Hierbei arbeiten das User Experience-Team (UX) und das Development-Team parallel, aber zeitlich versetzt. Das UX Team führt Usability Tests auf Basis des Mini-Release der vorherigen Iteration durch. Designs zur Implementierung für die nächste Iteration werden entworfen und validiert und Requirement Engineering Aktivitäten für den übernächsten Zyklus durchgeführt. Durch diesen Vorsprung gegenüber der Entwicklung, bietet sich die Möglichkeit Designs zu validieren, bevor sie implementiert werden. Es kann mehr Zeit in die Analyse und Anforderungsermittlung investiert werden. Dieses Prinzip wird in ähnlicher Weise auch von anderen Autoren aufgegriffen, wie beispielsweise Ambler (2007), Nummiaho (2006), Ungar & White (2008). Dabei ist es wichtig nicht „zu weit“ für die Zukunft zu designen, da während der Entwicklung immer wieder Änderungen aufkommen (Ferreira 2007).

2.5 Rollenmodell

Die Definition von verschiedenen Rollen innerhalb eines Projektteams zielt darauf, alle Kompetenzbereiche der Softwareentwicklung, mit entsprechen ausgebildeten Personen abzudecken.

In agilen Teams, wie sie in der Literatur beschrieben werden, gibt es keine feste Rollenzuteilung. Im Idealfall sind alle Teammitglieder in der Lage zu designen, zu programmieren und zu testen. Für Projekte bei denen UI und Interaktion nicht der wesentliche Erfolgsfaktor ist, mag das möglich sein. Andernfalls hat sich gezeigt, dass es notwendig ist, die entsprechenden Entwicklungsteams explizit um Personen mit Erfahrung im Bereich UI Design und/oder User Centered Design zu ergänzen, um die Gebrauchstauglichkeit sicherzustellen (Ferreira 2007; Lee 2006; Sy 2007). Kane hält es wiederum auch für möglich, dass Entwickler, mit einem Minimum an Training, leichtgewichtige „discount“ Usability Methoden anwenden können, wenn sie durch Interfaces Design Patterns, Style Guides und Evaluation durch Heuristiken ergänzt werden (Kane 2003).

2.6 Folgerungen für die Praxis

Resultierend aus der Betrachtung der Integrationsansätze lassen sich einige Kernaspekte identifizieren, welche die offensichtlich wesentlichen Punkte sind, die über ein erfolgreiches Zusammenspiel von UCD und agiler Softwareentwicklung entscheiden.

Eine Folgerung für die Analyse ist, dass genügend Wissen über die Nutzer vorhanden sein muss, bevor mit dem Design begonnen wird. Die Form der Analyse, also ob diese sehr umfangreich, vor Beginn eines Entwicklungsprojektes oder eher kurz am Anfang durchgeführt und dann inkrementell verfeinert wird, muss an die Komplexität der zu entwickelnden Software angepasst werden. Aus UCD Sicht ist ein ausreichend umfangreiches Basiswissen qualitätsentscheidend. Aus agiler Sicht ist es wichtig, dass zur Dokumentation leichtgewichtige Artefakte genutzt werden und ein Großteil des Wissens mit allen Teammitgliedern geteilt wird.

Usability Engineers und UI Designer sollten integraler Bestandteil der interdisziplinären Teams sein. Dies ist eine Grundvorrausetzung, damit übergeordnete Usability Anforderungen berücksichtigt, sowie das entsprechende Analyse-Aktivitäten (aus UE Sicht) richtig durchgeführt werden können. Für gewisse, einfache Usability Methoden können das auch Entwickler machen, die im Vorfeld darin unterrichtet werden und das nötige Bewusstsein dafür mitbringen. Das UI Design sollte ebenfalls von Personen mit Erfahrung und entsprechender Ausbildung entwickelt werden.

Es scheint sich bewährt zu haben, ein Team mit expliziten UCD- und UI-Design- Kenntnissen für die Analyse, Entwurf und Validierung bereitzustellen. Dieses Team erarbeitet Anforderungen

und Designs mit einem gewissen Vorsprung gegenüber der Implementierung. So können Designs vor der Entwicklung evaluiert werden, ohne den Entwicklungsprozess auszubremsen. Der Vorsprung sollte jedoch nicht zu groß sein, damit im Falle von Schwierigkeiten, z. B. bei der Implementierung, nicht zu viel Aufwand in spätere Änderungen investiert werden muss bzw. Konzepte entwickelt werden, die später keine Beachtung mehr finden.

Eine Folgerung bezüglich der Modellierung ganzheitlicher Arbeitsabläufe ist, dass in der Durchführung sichergestellt sein muss, dass nicht nur einzelne Arbeitsabläufe der Nutzer dokumentiert und implementiert, sondern auch die Beziehungen zwischen den Abläufen im konkreten Handlungskontext berücksichtigt werden. Die Verwendung von Artefakten, wie z. B. Use Cases, Szenarios, Task Maps und auch Persona-Beschreibungen stellen offensichtlich einen effektiven Weg dar, den Analysten, Produktmanagern und Nutzern ein ganzheitliches Bild von der Anwendung zu vermitteln und dadurch Anknüpfungspunkte und Problembereiche einfacher lokalisieren zu können.

Bezüglich des UI muss sichergestellt werden, dass Entwürfe mit Nutzern evaluiert werden, noch bevor sie implementiert werden. Um auf Änderungen und wechselnde Anforderungen schnell reagieren zu können, scheinen sich Papierprototypen als ideale Ergänzung zum Toolkit der agilen Softwareentwicklung erwiesen zu haben. Darüber hinaus müssen regelmäßige Usability Tests, mit allen Vertretern der Zielgruppe, im Prozess verankert werden.

Für die Analyse und das Testen ist die Zusammenarbeit mit echten Nutzern aus UCD Sicht unerlässlich. Deshalb muss sichergestellt werden, dass Endnutzer in der Gruppe der Customer Stakeholder vertreten sind und für diese Aktivitäten herangezogen werden können.

3.0 Anwendungsfall: Open Text's Agile Design

Dass sich die identifizierten Kernaspekte in der Praxis wiederfinden, wird an dem iterativen Entwicklungsprozess (IDP) von Open Text gezeigt.

Durch den iterativen Ansatz von Design und Entwicklung und eine enge Kundenbeteiligung in allen Phasen des Projektes, ist es möglich eine hervorragende Customer bzw. User Experience zu liefern. Dabei liefert eine frühe Customer Research Phase den (Nutzungs-) Kontext.

Der agile Prozess schafft den Rahmen für die Zusammenarbeit: festgelegte Feedback-Loops zwischen den externen Stakeholdern (Kunden, Endbenutzer) auf der einen Seite und den internen Beteiligten bei Open Text (Entwicklung, Produkt Management, Product Design, Sales und Services) erreichen, dass möglichst früh im Produkterstellungsprozess klar wird, was gebraucht wird und was nicht. Design-Entscheidungen werden iterativ getroffen, Kundenfeedback möglichst bis zur letzten Minute vor der eigentlichen Implementierung berücksichtigt.

3.1 IDP Artefakte

Folgende IDP Artefakte basieren auf Research Aktivitäten:

- **Vision**. Die High Level Sicht, das Ziel eines Releases.
- **Persona**.
- **Rock**. High Level Feature, typische Roadmap-Inhalte.
- **User Story**. Beschreibt eine Tätigkeit einer Persona und ihre zugrunde liegende Motivation.
- **Macro Story**. Beschreibt einen Schritt in oder ein Detail der User Story. Beispielsweise eine UX Spezifikation. Die technische Umsetzung wird von der Entwicklung in Form von **Micro Stories** spezifiziert.

Die Anwendung der Persona Methode von Alan Cooper garantiert eine äußerst benutzerfokussierte Entwicklung, da die Requirements (User Stories) aus dem Blickwinkel des Benutzers formuliert werden. Sowohl das User Interface Design als auch das User Interaction Design werden mit Hilfe diese Stories erläutert und definiert. Die für die Produkte erarbeiteten Persona Sets ergänzen die User Stories in idealer Weise.

3.2 IDP Projektstruktur

Das Projektteam ist in ein ‚Core Development Team' und ein erweitertes ‚Product Team' aufgeteilt. Dabei besteht das **Core Development Team** aus Product Designer, Poject Manager, Developer, Quality Engineer, Technical Writer. Das **Product Team** bezieht zusätzlich die externen und internen Stakeholder mit ein.

3.3 Iterativer Entwicklungsprozess

Der IDP bricht die Produktentwicklung in Iterationen von 2-4 Wochen Länge. Jede Iteration startet mit einem Planning Game und endet mit einem End Game. Landmarks fassen das Ergebnis mehrerer Iterationen zusammen, idealerweise bearbeiten sie wenigstens einen Rock. Landmarks werden wie ‚kleine interne Releases' behandelt. In Landmark Demos wird der aktuelle Stand vorgeführt; sie finden alle 2-3 Monate in einem größeren Rahmen vor dem gesamtem Produkt Team statt. Während der Landmark Demo erhält das Entwicklungsteam Feedback von allen Beteiligten. Die kurzen Zeitzyklen garantieren, dass das Feedback noch in der aktuellen Produktversion implementiert werden kann.

3.4 Agile Design-Methode

Die Ergebnisse der Research Aktivitäten durch die Product Designer liefern

das Fundament für das Design der User Experience für das laufende Projekt: Personas und Stories. Die Bedürfnisse und Aufgaben der Personas helfen, richtige Design-Entscheidungen im Hinblick auf die User Experience zu treffen und die Features zu priorisieren. Sie liefern zudem den Kontext um internen und externen Beteiligten die Software zu erläutern.

User Interface und User Interaction Design werden iterativ erarbeitet. Design-Entwürfe werden ausgewählten Stakeholdern (Design Partnern) präsentiert und von diesen bewertet. Deren Feedback dient als Grundlage für die nächste Design-Iteration. Auf diese Weise werden die Stakeholder involviert, bevor die eigentliche Implementierung begonnen hat bzw. möglicherweise schon erste Fehlentscheidungen getroffen wurden.

3.5 Verbesserte Usability

Die Integration von UCD Aktivitäten in allen Phasen des iterativen Entwicklungsprozess erhöht eine flexiblere Anwendung der Methoden und damit die Usability eines Produktes. Design-Iterationen während des gesamten Entwicklungsprozesses ermöglichen es, Feedback oder anderweitig gesammelte Erkenntnisse während der gesamten Entwicklungsphasen zu berücksichtigen. Durch die vorgeschaltete Research-Phase ist es garantiert, dass eine klare Vision formuliert wird, an der während der gesamten Projektlaufzeit kontinuierlich gearbeitet wird. Schließlich garantiert die intensive Kundenbeteiligung, dass die Features so entwickelt werden, wie sie die Endbenutzer wirklich brauchen.

4.0 Fazit

In dem vorliegenden Paper wurde das Wissen aus unterschiedlichen Integrationsansätzen des UCD und der agilen Softwareentwicklung angewandt, und deren wesentliche Gemeinsamkeiten als Kernaspekte identifiziert und beschrieben. Die resultierenden Folgerungen aus dem Vergleich wurden anhand eines Anwendungsfalles beschrieben und nun zur Diskussion gestellt. Die Folgerungen stellen keinen Anspruch auf Vollständigkeit, sondern sollen bei der Auswahl und Anwendung in der Praxis eine Hilfestellung bieten.

5.0 Literaturverzeichnis

Beyer , H.; Holtzblatt, K.; Baker, L. (2004): An Agile User-Centered Method: Rapid Contextual Design. In: Lecture Notes in Computer Science [Extreme Programming and Agile Methods - XP/Agile Universe 2004]. Heidelberg / Berlin: Springer Verlag.

Buxton, B. (2007) Sketching User Experiences: Getting the Design Right and the Right Design. Morgan Kaufmann Publ.

Chamberlain, S.; Sharp, H.; Maiden, N. (2006): Towards a framework for integrating agile development and user-centered design. Proc. In: Lecture Notes in Computer Science [Extreme Programming and Agile Processes in Software Engineering 2006]. Heidelberg / Berlin: Springer Verlag.

Cherry P. (2006): Questions that Sell: The Powerful Process for Discovering What Your Customer Really Wants. Mcgraw-Hill Professional.

Constantine, L. (2002): Process Agility and Software Usability: Toward Lightweight Usage-Centered Design. [WWW] http://www.foruse.com/articles/agiledesign.pdf [31.05.2008]

Cooper, A.; Reimann, R. (2007): About Face 3.0 The Essentials of Interaction Design. Indianapolis: Wiley Publishing Inc

Detweiler, M. (2007): Managing UCD within agile projects. In: Interactions Volume 14 SPECIAL ISSUE: UX Management (2007). New York: ACM Press.

Ferreira, J.; Noble, J.; Biddle, R. (2007) Up-Front Interaction Design in Agile Development In: Lecture Notes in Computer Science [8th International Conference on Agile Processes in Software Engineering and eXtreme Programming]. Heidelberg / Berlin: Springer Verlag

Kane, D. (2003): Finding a place for discount usability engineering in agile development: throwing down the gauntlet. In: Proceedings of the Agile Development Conference 2003. IEEE Computer Society

Norman, D. (2006): Why doing user observations first is wrong. In: Interactions Volume 13 (2006). New York: ACM Press.

Patton, J. (2008). Getting Software RITE. In: Software, IEEE 2008 Volumen 25. IEEE Computer Society

Patton, J. (2002) Designing Requirements: Incorporating Usage-Centered Design into an Agile SW Development Process. In: Lecture Notes in Computer Science [Extreme programming and agile methods: XP/Agile Universe 2002]. Heidelberg / Berlin: Springer Verlag.

Pruitt, J.; Adlin, T. (2006): The Persona Lifecycle - Keeping People in Mind Throughout Product Design. Morgan Kaufmann Publishers

Sy, D. (2007): Adapting Usability Investigations for Agile User-Centered Design. Journal of Usability Studies

Ungar, J. M.; White, J. A. (20080: Agile user centered design: enter the design studio-a case study. In: CHI '08 extended abstracts on Human factors in computing systems.New York: ACM Pres

Wolkerstorfer, P.; Tscheligi, M.; Sefelin, R.; Milchrahm, H.; Hussain, Z.; Lechner, M.; Shahzad, S. (2008): Probing an agile usability process. In: CHI '08 extended abstracts on Human factors in computing systems.New York: AC