

Agile nutzerzentrierte Softwareentwicklung mit leichtgewichtigen Usability Methoden – Mythos oder strategischer Erfolgsfaktor?

Thomas Memmel
Mensch-Computer Interaktion
Universität Konstanz
Universitätsstrasse 10, Box 73
78457 Konstanz
<http://hci.uni-konstanz.de>
memmel@inf.uni-konstanz.de

Henning Brau
Daimler AG
Data and Process Management
Team Psychology in Engineering
Wilhelm-Runge-Strasse 11
89013 Ulm
henning.brau@daimler.com

Dirk Zimmermann
Eigelstein 27
50668 Köln
dirk.zimmermann@gmail.com

Abstract

In diesem Beitrag stellen wir Möglichkeiten zur Verzahnung von Usability Engineering und agiler Softwareentwicklung vor. Dabei beleuchten wir zunächst die Gemeinsamkeiten und Unterschiede beider Vorgehensweisen. Hinsichtlich einer dringend erforderlichen Best Practice für eine agile, nutzerzentrierte Softwareentwicklung erläutern wir Möglichkeiten für ein Down-Stripping von Usability Engineering Techniken. Schließlich schlagen wir konkrete Strategien zur

Integration von leichtgewichtigen Usability Engineering Methoden in agile Entwicklungsprozesse vor. Ziel eines Hybrid-Vorgehensmodells muss dabei weiterhin die Entwicklung von User Interfaces mit hoher Usability und User Experience Qualität sein. Daher müssen Methoden beider Welten mit Bedacht ausgewählt, verschmolzen und intelligent im iterativen Entwicklungszyklus eingesetzt werden. Erst dann kann trotz knapper Budgets und ambi-

tionierter Zeitplanung eine hohe User Interface Qualität sichergestellt werden. IT-Organisationen, die den Brückenschlag der Disziplinen schaffen und in der methodischen Praxis umsetzen, können so einen strategischen Wettbewerbsvorteil erreichen.

Keywords: Agile, User Interface, Usability Engineering

1.0 Einleitung

Die Wichtigkeit des User Interface (UI) wird bereits in vielen zukunftsorientierten Entwicklungs- und Fachabteilungen erkannt. Daher erhalten grundsätzliche Methoden des Usability Engineering ebenso Einzug in Entwicklungsprozesse wie eine Sensibilisierung für das UI Design. Gleichzeitig herrscht jedoch ein enormer Zeit- und Kostendruck vor. Produktentwicklungszeiten werden verkürzt, Softwaresysteme müssen günstiger hergestellt werden und deren Entwicklung wird daher zunehmend auch ins Ausland verlagert. In diesem Kontext besteht die Gefahr, dass Usability Engineering Methoden auch schnell wieder in den Hintergrund rücken.

Zahlreiche Faktoren können dazu beitragen, dass Softwareprojekte angesichts dieses enormen Spannungsfelds scheitern oder nur mit unzureichender

UI Qualität abgeschlossen werden. Dazu gehören:

1. Die Anforderung trotz straffer Rahmenbedingungen gebrauchstaugliche Softwareprodukte zu entwickeln.
2. Der Anspruch eine hohe UI Qualität zu gewährleisten, die der Organisation auch eine strategisch wichtige User bzw. Customer Experience sicherstellt.
3. Der Einfluss von Akteuren aus weitestgehend separierten Disziplinen, darunter vor allem Softwareentwickler, Usability Experten und Business Analysten.
4. Das Fehlen geeigneter Methoden und Vorgehensmodelle, die solchen Herausforderungen angemessen Rechnung tragen.

Agile Methoden erfreuen sich in diesem Kontext einer zunehmenden Be-

liebtheit bei der Entwicklung von Softwaresystemen. Auch für Usability Experten, deren angewandte Methoden nach wie vor von vielen Softwareentwicklern als zeit- und kostenintensiv angesehen werden, sind agile Methode sehr interessant. Jedoch müssen einige Mythen, Vorurteile und Hindernisse überwunden werden, wenn eine Hybrid-Vorgehensweise im Sinne von *Agile Usability Engineering* eingeführt werden soll.

2.0 Bist Du schon agil oder dokumentierst Du noch?

Durch eine inkrementelle Implementierung, zahlreiche Iterationen und schnelle Release-Zyklen wird mit agilen Methoden eine deutliche Flexibilisierung bei gleichbleibend hohen Qualitätsansprüchen ermöglicht. Auf diese Weise kann eine deutlich kürzere Time-to-Market erreicht werden. In keinsten Weise korrekt ist jedoch die immer noch weit verbreitete Annahme, dass damit eine

Eliminierung jeglicher Dokumentation verbunden ist.

Die Motivation hinter den agilen Modellen unterscheidet sich sicherlich von klassischen Modellen in vielerlei Hinsicht. Es gibt keine monolithischen Phasen und natürlich wird auch weniger Wert auf eine umfangreiche Anfertigung unterschiedlicher Artefakte gelegt. Dabei kann das „Einsparen“ jedoch vor allem auf leitende agile Prinzipien zurückgeführt werden, darunter *Model with a purpose* oder *Travel Light*. Diese Maximen sehen beispielsweise vor, nur unbedingt erforderliche Modelle aufzubauen und zu pflegen. Vielmehr als in anderen Entwicklungsmethoden kommt es gleichzeitig auch auf Kommunikation und die Interaktion aller Projektbeteiligten an (Schwaber 2004).

Entwicklungsmodelle des Usability Engineering und agile Methoden, wie z.B. eXtreme Programming (XP), verfolgen im Grunde ein und dieselbe Zielstellung: Leistungsstarke und stabile Anwendungen mit hohem Nutzwert und Anwenderakzeptanz hervorzubringen. Um solche Applikationen erfolgreich zu entwickeln, müssen unabhängig vom Entwicklungsvorgehen die Bedürfnisse der späteren Nutzer im Anwendungskontext verstanden werden.

Mitunter befürchten Usability Experten jedoch, dass im Mahlstrom der engen Release Planung, die vornehmlich auf die Erzeugung lauffähiger Features abzielt, die konkreten Nutzeranforderungen zu wenig Beachtung erhalten. Die Erfolge agiler Softwareprojekte beziehen sich tatsächlich oft auf Systeme, bei denen die Entwicklung und Qualität des UI nicht im Vordergrund stehen. Derartige Bedenken müssen analysiert und verstanden werden, wenn über Agile Usability Engineering nachgedacht wird.

3.0 Der gemeinsame Nenner

Der Blick auf wichtigste Prinzipien und Praktiken von XP oder Agile Modeling (AM) zeigt, dass es viele Gemeinsamkeiten gibt (Gundelsweiler et al. 2004; Memmel et al. 2007; Nebe et al. 2007).

So kennt XP mit Story Cards und Task Cards ähnliche Artefakte wie das Usability Engineering mit User Profiles oder Task Models. Das grundsätzliche Erfordernis von XP Prozessen, in häufigen Zyklen schnell vorzeig- und testbare Ergebnisse zu erzielen, ist mit Usability Prozessen vereinbar. Hier sind ebenfalls frühe Prototypen vorgesehen, um Feedback in die weitere UI Entwicklung zu integrieren. Die dazu notwendige Einbeziehung von Stakeholdern als permanente Quelle für neue Anforderungen und Feedback ist dem aus im Usability Engineering bekannten Prinzip des kollaborativen Designs ähnlich (Gundelsweiler et al. 2004).

Sicherlich existieren neben den leicht feststellbaren Übereinstimmungen auch grundlegende Konflikte. So wird der agile Entwicklungsprozess in kleine, überschaubare Problemstellungen zerlegt (Small Releases) und zügig inkrementell ausgeführt. Dabei ist es nicht erforderlich, dass alle Anforderungen bereits zu Beginn verstanden und erfasst worden sind. Während der Projektlaufzeit muss es zudem möglich sein, Anpassungen in verschiedenem Umfang vorzunehmen (Refactoring). Im Gegensatz zu dieser Philosophie wird im Usability Engineering zu Beginn relativ viel Zeit für die Anforderungsermittlung und die UI Konzeption genutzt. In die eigentliche Entwicklung soll erst eingestiegen werden, wenn genügend Wissen bezüglich des Nutzungskontextes vorhanden ist und eine vollständige Architektur des UI entwickelt worden ist. Ein permanen-

tes Redesign des UI kann schnell zu massiven Benutzungsproblemen führen. So würde eine grundsätzliche Veränderung des Navigationskonzeptes, der Organisation der Informationspräsentation oder der gewählten Interaktionstechniken von Small Release zu Small Release dazu führen, dass grundlegende Dialogprinzipien wie Erwartungskonformität, Selbstbeschreibungsfähigkeit oder Erlernbarkeit verletzt würden (Gundelsweiler et al. 2004).

In agilen Methoden ist es auch nicht zwangsläufig der Endbenutzer, der während der kurzen Anforderungsanalyse zu Beginn befragt wird (Mommel et al. 2007; Nebe et al. 2007). So können sich einige offene Fragen bei der Erstellung von User Stories durch Dritte ergeben. Ähnliches gilt für abschließend durchgeführte Akzeptanztests, die nicht etwa die Bewertung durch Endnutzer erhebt, sondern die Erfüllung der vereinbarten Projektumfänge mit dem Auftraggeber. Somit vernachlässigen die agilen Softwareentwicklungsprozesse aus Sicht des Usability Engineering eine Reihe wichtiger Vorgehensweisen zur Entwicklung gebrauchstauglicher Software.

4.0 Flexible Rollenverteilung

Eine Grundannahme agiler Methoden ist, dass die Projektmitglieder vergleichbare Qualifikationen haben und somit in einem homogenen Team entwickelt wird. Dies lässt sich schlecht mit Usability Engineering vereinbaren: In DIN EN ISO 13407 für nutzerzentrierte Entwicklung werden explizit multidisziplinäre Teams eingefordert, in denen neben Programmierern auch Designer, Stakeholder und Usability Experten vertreten sein sollen. Zudem sind Usability Experten, die zusätzlich über profunde Softwareentwickler Kompetenzen verfügen eher selten. Gleiches gilt für den umgekehrten, eher noch exotischeren

Fall. Traditionelle Rollen müssen daher insgesamt durchbrochen werden, so dass beispielsweise auch Entwickler am Designvorgang beteiligt sind und Usability Experten an der Architekturfestlegung. Gerade wenn sich das Agile Usability Engineering an XP anlehnt, kann die interdisziplinäre Vermischung der Akteure im Rahmen von Pair Programming erfolgen (Gundelsweiler et al., 2007). Die offene Kooperation selbst ermöglicht dann durch das konstruktive (auch soziale) Miteinander der verschiedenen Disziplinen und Perspektiven eine gegenseitige Annäherung.

Kernkompetenzen der multidisziplinären agilen Projektarbeit

Anforderungen an Entwickler:

- UI-Kompetenzen aufbauen – Grundlagen, Prinzipien und Methoden des Usability-Engineerings sollten ihnen zumindest bekannt sein.
- Usability als gewichtigen Qualitätsfaktor verstehen – Qualitätsabsicherung ist ein wichtiger Punkt der Agilen Systementwicklung. Für viele Systemeigenschaften gibt es definierte Absicherungsmechanismen. Warum also nicht für Usability?
- Styleguides respektieren – Gute Programmierer sind gewöhnt, den Konventionen der Programmierung und Systemarchitektur zu folgen, gleiches sollte auch für die Einhaltung der Styleguides bei der UI-Gestaltung gelten.

Anforderungen an Usability Experten:

- Kompetenzen in der Agilen Systementwicklung aufbauen – Grundlagen, Prinzipien und Methoden der agilen Systementwicklung sollten ihnen zumindest bekannt sein.
- Im Entwicklungsteam positionieren – Anstelle der Trennung zwischen Entwicklungs-, Design- und Usability-Aufgaben sind stetige Kommunikation und Kooperation notwendig.
- Vorurteile abbauen – Selbst das viel gescholtene XP ist weitaus strukturierter und auch anforderungsorientierter als sein Ruf. Die agilen Methoden sind adaptierbar an die jeweiligen aktuellen Bedarfe, ebenso die Usability-Engineering Methoden.

5.0 Agile Usability Engineering

Aufbauend auf den bisherigen Überlegungen können für die unterschiedlichen Entwicklungsphasen konkrete Methoden für ein Agile Usability Engineering herauskristallisiert werden. Im folgenden Beispiel orientieren wir uns dazu am XP Prozess ergänzt um Anteile aus der Scrum Methodik und erweitern diese um Bestandteile des Usability Engineerings. Zeitintensive Maßnahmen haben dabei keinen Platz, so dass tiefgehende Verhaltens- und Prozessanalysen von Nutzern und Nutzungskontext, langwierige Interviewserien oder auch aufwändige Evaluationsverfahren kaum in Betracht kommen.

5.1 Anforderungsphase (Iteration 0)

In dieser Phase ist der direkte Kontakt von Entwicklern und Stakeholdern für den Gesamterfolg der Entwicklung sehr hilfreich. Methoden, die sich für eine Nutzeranalyse empfehlen sind beispielsweise offene Interviews, Fokusgruppen, teilnehmende Beobachtungen, Strukturlegetechniken (Card Sorting) sowie das Aufzeichnen von Storyboards und Personas zusammen mit den Stakeholdern.

Der gewonnenen Anforderungen können sehr gut mit Role Models und Role Maps (Constantine & Lockwood, 1999) festgehalten werden. Dokumentiert werden sollen die für die Anwendung wichtigen Charakteristika der Rollen, sowie erforderliche Interaktionsschemata. Anschließend werden die Benutzerrollen mit Prioritäten versehen (Focal User Roles) und nach Relevanz für den Produkterfolg sortiert.

Essentiell sind auch die Aufgaben der Benutzer, die durch Szenarien, Essential Use Cases und Use Case Maps (Constantine & Lockwood, 1999) oder auch User Stories (Cohn, 2004) beschrieben werden können. Wie schon die Benutzerrollen werden auch die Task Cases sortiert, wobei man sich an der aus XP (Beck, 1999) bekannten Einteilung orientieren kann (Required - do first, Desired - do if time, Deferred - do next time). Ergänzt wird diese Einteilung um essentielle Task Cases, die besonders komplex oder neuartig sind. Diese werden zu Szenarien ausformuliert, um das Verständnis im Team besser zu kommunizieren. Um Dokumentationsarbeit einzusparen reicht es aus, die ausformulierten Szenarien direkt auf einer Indexkarte festzuhalten. Sind die Aufgaben alle erfasst, ist es möglich, Cluster zu bilden, die innerhalb eines Anwendungsbereichs wichtige Programmteile aufzeigen.

Der wesentliche Vorteil der Role Models/Role Maps und Essential Uses Cases/Use Case Maps aus agiler Modellierungssicht besteht darin, dass sie gut zur agilen Philosophie passen (z.B. Nutzung von Indexkarten, Fokussierung auf das Essentielle).

Es hat sich ebenfalls für agiles UI Design bewährt, verschiedene Ergebnisse des Usability Engineerings jeweils als einzelne Anforderungen zu dokumentieren (Nebe et al., 2007). Aus den Ebenen „Nutzungskontext“, „Nutzungsanforderungen“ und „Konzeption/Design“ lassen sich so sowohl allgemeine Usability-Anforderungen modellieren (Effektivität, Effizienz und Zufriedenheit), wie auch Workflow-Anforderungen (im Sinne der ISO 9241 Dialogprinzipien) und nicht zuletzt Anforderungen des UI, entsprechend den Informationsdarstellungskriterien der DIN EN ISO 9241 (Zimmermann & Groetzbach, 2007) modellieren. Durch die feine Granularität lassen sich einzelne Anforderungen leicht auf mehrere Teams aufteilen, stellen aber gleichzeitig durch ihre Ableitung aus und ihren Bezug zu den Analyseartefakten eine verlässliche Richtschnur für die Evaluation in der verteilten Entwicklung dar.

Bei agilen Ansätzen werden Design Prinzipien und Style Guides kaum erwähnt. Vor allem in größeren Firmen mit ausgeprägter Corporate Identity (CI) existieren oft umfassende firmenweit geltende Style Guides, die nicht nur das Look and Feel, sondern auch die CI-spezifischen Aspekte des UI regeln. Als Alternative zur Erstellung eines Style Guides schlagen wir eher einen Prototyping-getriebenen Spezifikationsprozess für das UI vor, bei dem die wichtigsten Informationen zum System zusammen mit dem UI Design als Simulation bereitgestellt werden (Mommel et al. 2008). Durch sogenannte interaktive UI Spezifikationen kann auch die Zusammenarbeit mit einem (internen oder externen) Zu-

lieferer verbessert werden, der in vielen Fällen für die Implementierung verantwortlich ist.

5.2 Konzeptionsphase

In der Anforderungsphase wird bereits die Grundlage für eine Aufteilung von Funktions- und UI-Entwicklung gelegt, in der ab der Konzeptionsphase die gemeinsamen Ansätze unabhängig voneinander ausdifferenziert werden. Dies ist vor allem bei größeren Projekten sinnvoll. Voraussetzung ist allerdings, dass sich die Programmierung einfach in Funktionalität und UI aufteilen lässt. Eine solche Trennung kann mit Hilfe von Software Patterns erreicht werden. Es ist jedoch Fakt, dass der Einfluss der Benutzeroberfläche auf die Architektur mit steigendem Maß an Interaktivität zunimmt. Daher ist eine enge Zusammenarbeit und laufende Abstimmung dieser Stränge unerlässlich, um die spätere Systemintegration reibungslos zu ermöglichen. Dazu muss eine minimalistische UI-Spezifikation erstellt werden (Constantine & Lockwood 2002), die den Entwicklern der Systemarchitektur als Vorlage für die Schnittstellen der abhängigen Komponenten dient.

Insbesondere für den UI-bezogenen Entwicklungszweig sollten Prototypen erzeugt oder weiterentwickelt werden. Papierprototypen können für das schnelle Testen erster Ideen genutzt werden, während elaboriert den Workflow abbildende Prototypen (beispielsweise in Adobe Flash erstellt) ideal für Präsentationen vor Publikum und die Simulation der Interaktion sind. Die Erstellung von realistischen Prototypen ist zwar aufwändiger, diese können jedoch in der Konstruktionsphase weiterverwendet werden und als Spezifikation dienen (Mommel et al. 2008).

Auch erste Evaluationen können schon in der konzeptionellen Phase

durchgeführt werden. Agile Usability Techniken, die mit wenig Aufwand eingesetzt werden können sind, zum Beispiel User und Expert Reviews sowie Usability Inspektionen. Durch eine Beteiligung von Stakeholdern kann die Kooperative Heuristische Evaluation (Brau 2004) die Nutzer- und Expertensicht zusammenführen. Dazu werden realistische Anwendungsszenarien für den zu inspizierenden Funktionsumfang entwickelt. Jeweils ein Usability Experte bearbeitet die Anwendungsszenarien zusammen mit einem Nutzer. Dieser wird aufgefordert die jeweiligen Handlungsschritte zu kommentieren, bzw. der Usability Experte stellt ihm Verständnisfragen zum realen Nutzungskontext. Dies hilft dem Usability Experten, die Sichtweise der Nutzer zu verstehen und entsprechend seine eigenen Ergebnisse auf die Nutzeranforderungen auszurichten. Auch für Entwickler kann es sehr aufschlussreich sein, an einer solchen Sitzung als stiller Beobachter teilzuhaben.

5.3 Konstruktionsphase

Bevor die Programmierung beginnen kann, ist ein Iteration Planning (XP) oder Sprint Planning (Scrum) empfehlenswert. Neben den funktionalen Unit- bzw. Acceptance-Tests (XP), die später automatisiert ablaufen können, können jetzt auch weitere Usability Testfälle für das ausgereifte System (z.B. anhand möglicher Testaufgaben) überlegt werden, sowie Evaluationskriterien und Zielgrößen für die User Acceptance Tests im Sprint Review innerhalb des Scrum Ansatzes.

In dieser Phase findet die teilweise parallel ablaufende Implementierung des UI und der Systemarchitektur statt. UI-Komponenten, die für die Implementierung der Systemarchitektur schnell angeliefert werden müssen, können zunächst mit geringer Genauigkeit umgesetzt und in späteren Iterationen verfei-

ner werden. Dies ist unkritisch, solange das konzeptuelle Modell nicht verändert wird.

Wie auch in der Konzeptionsphase können Usability Studien angewendet werden, um die Gebrauchstauglichkeit des UI sicherzustellen. Funktionstest sollten parallel verlaufen, um Zeit einzusparen. Testergebnisse können auf sogenannten *Defect Cards* (Gundelsweiler et al., 2004) festgehalten, priorisiert und die Behebung für folgende Iterationen eingeplant werden.

6.0 Übergreifende Koordination

Um gerade in größeren Organisationen, in denen viele kleinere Teams parallel arbeiten, die Kohärenz und Konsistenz der Lösung zu gewährleisten, hat sich parallel zu den XP-Ansätzen innerhalb der Scrum-Methodik (Schwaber, 2004) für teamübergreifende Themen das Konzept der „Scrum of Scrums“ bewährt (Larman, 2004). Während die Scrum-Teams in kurzen Inkrementen mit täglichen Abstimmungstreffen und schnellen Iterationen die eigentliche Lösung entwickeln, treffen sich Vertreter jedes der Scrum-Teams in größeren Anständen zu speziellen übergreifenden Themen, wie etwa der Architektur, Datenbankmodellierung, usw.

Ein „Usability Scrum of Scrums“ greift diese Methodik auf und hilft den in einzelnen Teams verteilten UI Designern, ihre Ergebnisse regelmäßig miteinander abzugleichen, Evaluationsergebnisse auszutauschen und Ideen und Modelle für Folgethemen zu generieren. So werden die Ideen aus der Analysephase auch durch das gesamte Projekt in regelmäßigen Abständen überprüft und gegebenenfalls modifiziert, jedoch nicht lokal, sondern über die einzelnen Teams hinweg.

7.0 Zusammenfassung

Agile Prozesse und Usability Engineering haben grundlegend abweichende Herangehensweisen an das Software Engineering. Beide sehen aber den Einsatz von Methoden zur Nutzer- und Kontextanalyse vor. Agile Methoden erfordern insgesamt ein deutlich flexibleres und schlankeres Usability Engineering, Für eine Hybrid-Vorgehensweise ist ein hohes Maß an multidisziplinärer Kooperation und organisationalem Umdenken notwendig. Gelingt eine strukturierte Verknüpfung der beiden Entwicklungsparadigmen, können viele Prozesse parallelisiert werden, so dass in kürzerer Zeit gebrauchstaugliche Anwendungen entwickelt werden können.

Im Vergleich zum ursprünglichen XP und Scrum Prozess führen die im Beitrag skizzierten Methoden sicherlich zu weniger „Agilität“. Dafür ermöglicht die Erweiterung aber, eine gebrauchstaugliche Software zu entwickeln. Insgesamt erscheint es uns aber noch mehr als gerechtfertigt, von einem agilen Hybrid-Vorgehensmodell zu sprechen.

8.0 Literaturverzeichnis

Brau, H. & Schulze, H. (2004): Kooperative Evaluation: Usability Inspektion in komplexen und verteilten Anwendungsdomänen. In: Peissner, M. & Hassenzahl, M. (Hrsg.). Usability Professionals 2004. Fraunhofer IRB Verlag: Stuttgart.

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

Constantine, Larry L. (2002): Process Agility and Software Usability: Toward Lightweight Usage-Centered Design. Information Age.

Constantine, Larry L.; Lockwood, Lucy A. D. (1999): Software for Use: A Practical Guide to the Methods of Usage-Centered Design. Addison-Wesley Professional.

Gundelsweiler, Fredrik; Memmel, Thomas; Reiterer Harald (2004): Agile Usability Engineering. Proceedings of the 4th Mensch & Computer conference (MCI 2007, Paderborn, Germany), In: Keil-Slawik, R.; Selke, H.; Szwillus, G.: Mensch & Computer 2004: Allgegenwärtige Interaktion, Oldenbourg Verlag, München, S. 33-42

Larman C. (2004). Agile & Iterative Development – A Manager's Guide, Addison-Wesley: Boston.

Mommel, T.; Gundelsweiler, F.; Reiterer, H. (2007): Agile Human-Centered Software Engineering. In Proc. of 21st BCS HCI Group conference (HCI 2007, UK), in: Linden J. Ball, M. Angela Sasse, Corina Sas, Thomas C. Ormerod, Alan Dix, Peter Bagnall and Tom Mc Ewan: "HCI...but as we know it", British Computer Society, S. 167-175

Mommel, T.; Geis, T.; Reiterer, H. (2008): Methoden, Notationen und Werkzeuge zur Übersetzung von Anforderungen in User Interface Spezifikationen. In: Brau, H., Diefenbach, S., Hassenzahl, M., Koller, F., Peissner, M. & Röse, K. (Hrsg.). Usability Professionals 2008. Fraunhofer IRB Verlag: Stuttgart.

Nebe, K., Düchting, M. & Zimmermann, D. (2007). Integration von User Centred Design Aktivitäten in Agile Softwareentwicklung. In: H. Brau & K. Röse (Hrsg.): Usability Professionals 2007. Fraunhofer IRB Verlag: Stuttgart.

Schwaber, K. & Beedle, M. (2002). Agile Software Development with Scrum. Prentice Hall: Upper Saddle River.

Schwaber, K. (2004): Agile Project Management with Scrum. Redmond: Microsoft Press.

Zimmermann, D. & Groetzbach, L. (2007). A Requirement Engineering Approach to User Centered Design. In: Proceedings of the 2007 HCI International Conference (Beijing, P.R. China, July 22-27, 2007)..