Alexander Mädche, Dieter Wallach

Agile und Nutzerzentrierte Softwareentwicklung

Einleitung

In den letzten Jahrzehnten entwickelte sich die mechanistische Sicht der Softwareentwicklung von einem phasenbasierten und sequentiellen Ansatz hin zu einem dynamischen Prozess mit iterativen Zyklen. Agile Softwareentwicklung wie sie beispielsweise durch das Vorgehensmodell SCRUM propagiert wird, konnte sich in den letzten Jahren in der Praxis schnell und mit nachweisbar positiven Effekten auf die Ergebnisse von Softwareentwicklungsprojekten etablieren.[1] Agile Entwicklungsmodelle betonen die Bedeutung von unvorhersehbaren Veränderungen im Sinne der Kundenorientierung und richten den primären Fokus auf die Entwicklung von nützlicher Software, d.h. Software, welche für den Kunden konkreten Nutzen stiftet. Die Gebrauchstauglichkeit (Usability) wird im Software Engineering üblicherweise als nicht-funktionale Anforderung betrachtet und nicht explizit in den Vordergrund gestellt. Im interdisziplinären Bereich Human-Computer Interaction (HCI) wurden relativ losgelöst vom Bereich des Software Engineering verschiedenste Vorgehensmodelle zur Erstellung gebrauchstauglicher Software entwickelt. Ein wichtiger Vertreter dieser Modelle ist die nutzerzentrierte Gestaltung (User-Centered Design, UCD), welche die Ziele und Bedürfnisse der Endbenutzer einer Software in den Entwicklungsfokus stellt.[2] Eine iterative Verfeinerung der zu entwickelten Lösung mit einer kontinuierlichen Integration von Endbenutzern soll sicherstellen, dass eine Software entsteht, welche den Benutzer effizient, effektiv und zufriedenstellend bei der Aufgabenbearbeitung unterstützt.

Mit der wachsenden Bedeutung einer hohen Gebrauchstauglichkeit von Software stellt sich die Frage nach Ähnlichkeiten und Unterschieden existierender Ansätze zur agilen Softwareentwicklung im Vergleich zu Vorgehensmodellen der nutzerzentrierten Entwicklung – und wie sich diese gegebenenfalls integrieren lassen. Um diese Frage zu beantworten, wurden im Rahmen einer umfassenden Literaturstudie relevante Publikationen aus den beiden genannten Entwicklungsansätzen identifiziert, systematisch analysiert und auf dieser Grundlage fünf grundlegende Prinzipien zur agilen und nutzerzentrierten Softwareentwicklung abgeleitet. Nach einer kurzen Vorstellung der Literaturstudie beschreibt dieser Beitrag die identifizierten Prinzipien jeweils im Überblick.

Methodische Vorgehensweise in der Literaturstudie

Die Literaturstudie wurde nach etablierten wissenschaftlichen Methoden durchgeführt.[3] In einem ersten Schritt wurden eine Suchstrategie definiert sowie Selektionskriterien und -prozeduren dokumentiert. Existierende Arbeiten wurden auf die zentralen Bereiche des Software Engineering,

1 Vgl. Dybå & Dingsøyr, 2008.
2 Vgl. Norman, 1988.
3 Vgl. Kitchenham, 2007.

AND	OR	Ergonomics, Human-Computer Interaction, Computer-Human Interaction, Interaction Design, Usability, User Experience, User-Centered Design, UI Design, Interface Design
	OR	Agile, Scrum, Extreme Programming, Lean, Crystal Clear, Feature Driven Development, Dynamic Software Development
	OR	Software Development, Systems Development

Abbildung 1: Suchanfrage

Human-Computer Interaction sowie der Wirtschaftsinformatik fokussiert. Die in Abbildung 1 dargestellte Suchanfrage wurde formuliert und auf die für den Gegenstandsbereich relevanten Literaturdatenbanken (ProQuest, Elsevier ScienceDirect, EBSCO Host, EEE Xplore, ACM Digital Library, Springer Link) angewendet.

Ein auf den Ergebnissen der Suche basierender Selektionsprozess setzte sich aus vier Phasen zusammen. In Phase 1 und 2 wurden die Suchergebnisse der Datenbanken integriert und Duplikate eliminiert, dies resultierte in 1034 Publikationen. In Phase 3 wurden die Titel und Zusammenfassungen der Publikationen analysiert und alle Beiträge, welche nicht auf agile bzw. nutzerzentrierte Entwicklung fokussierten, eliminiert. Nach einer Vorwärts-/Rückwärtssuche wurden insgesamt 148 Publikationen in die engere Wahl gezogen. Nach einer weiteren Qualitätsprüfung in Phase 4 wurden 83 Publikationen in den Analyseprozess aufgenommen.

Abbildung 2 stellt den zeitlichen Verlauf der Publikationen dar. Aus dem Verlauf wird ersichtlich, dass bereits seit fast einem Jahrzehnt ein Interesse an agiler und nutzerzentrierter Softwareentwicklung existiert. Seit 2005 wurde mit Ausnahme von 2011 in den betrachteten Quellen eine zweistellige Anzahl von Publikationen pro Jahr im genannten Themenbereich identifiziert.

Zur Analyse der Publikationen wurden folgende vier Integrationskategorien als Ausgangspunkte eingesetzt: Prozesse, Praktiken, Menschen/Soziale Aspekte und Werkzeuge.[4] Basierend auf diesen vier Kategorien wurde sukzessive ein verfeinertes hierarchisches Kodierungsschema entwickelt. Dabei wurden beispielsweise konkrete Praktiken (Prototyping, Personas, etc.) identifiziert und in das Kodierungsschema als Detaillierung der Kategorie Praktiken aufgenommen. Jede Publikation wurde gelesen, im Detail analysiert und auf der Ebene von Textpassagen kodiert. Zur Identifikation von Prinzipien wurden die Häufigkeit von Kodierungen sowie der Kontext der jeweiligen Publikationen berücksichtigt.

Fast die Hälfte der identifizierten Beiträge setzte einen Fokus auf konkrete Praktiken, knapp 30% fokussierten auf Prozesse. Die zahlreichen Publikationen im Bereich Praktiken und Prozesse bedeuten jedoch nicht, dass hier bereits fest etablierte Konzepte vorliegen. Tatsächlich fand sich eine Vielzahl von teilweise sogar widersprüchlichen Vorgehensmodellen zur Adressierung einer agilen und nutzerzentrierten Softwareentwicklung. Menschen und Werkzeuge stehen mit den verbleibenden 20% eher im Hintergrund der analysierten Forschungsarbeiten in den untersuchten Disziplinen. Auf Grund der starken Betonung von Prozessen und Praktiken in den existierenden Publikationen wurden auf Basis der vorliegenden Daten ausschließlich Prinzipien für diese beiden Bereiche identifiziert. Der nachfolgende Abschnitt fasst die angesprochenen Prinzipien zusammen.

4 Vgl. Barksdale & McCrickard, 2012.

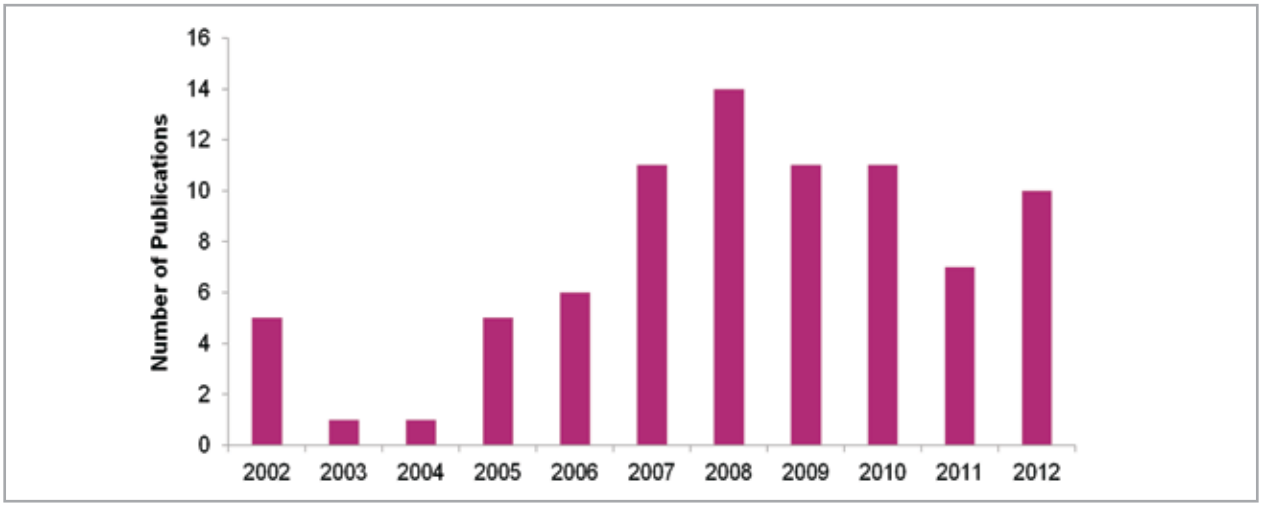

Abbildung 2: Zeitlicher Verlauf

Prinzipien zur Agilen und Nutzerzentrierten Softwareentwicklung

Die fünf grundlegende Prinzipien zur agilen, nutzerzentrierten Softwareentwicklung können entlang der beiden Kategorien Prozesse und Praktiken klassifiziert werden, wobei drei Prinzipien prozessuale Aspekte abdecken und die beiden verbleibenden Prinzipien primär auf Praktiken fokussieren.

Prinzipien zu Prozessen

Die Prozessperspektive fokussiert auf die übergreifende Orchestrierung der einzelnen Aktivitäten im Softwareentwicklungsprozess. Abbildung 3 stellt die Kodierungen in der Prozesskategorie und deren Häufigkeit der Nennung in den identifizierten Publikationen dar. Die Kategorien lassen sich in die drei Prinzipien „Getrennte Exploration und Erstellung“, „Iterative und Inkrementelle Erstellung“ und „Parallele, Synchronisierte Prozesse“ organisieren. Diese werden im Folgenden konkreter beschrieben.

Prinzip I: Getrennte Exploration und Erstellung

Das Prinzip I basiert auf einer Zusammenfassung der Kodierungen „Design Up Front“, „Cycle Zero“, „Cohesive Overall Design“, „Product Vision/Innovation“. Die grundlegende Annahme von Prinzip I ist, dass einer agilen, nutzerzentrierten Softwareentwicklung immer eine Explorationsphase notwendig vorangestellt werden muss. Hierbei sind unterschiedliche Begrifflichkeiten und Konzepte im Software Engineering bzw. der HCI festzustellen. So wird beispielsweise im agilen Softwareentwicklungsmodell SCRUM häufig der sogenannte Sprint bzw. Cycle Zero zur Analyse und Verfeinerung von Anforderungen genutzt. Up-Front Design Aktivitäten dienen der Problem- und Bedürfnisanalyse der Nutzer und unterstützen dabei die Fokussierung des Entwicklungsprozesses. Die Kategorie „Product Vision/Innovation“ geht noch einen Schritt weiter und betont das Potenzial der nutzerzentrierten Gestaltung zur Identifikation von Innovationsmöglichkeiten sowie zur Kommunikation der Produktvision in das Entwicklungsteam.

Prinzip II: Iterative und inkrementelle Erstellung

Das Prinzip II basiert auf einer Zusammenfassung der beiden Kategorien “Iterative Design/Development“ und „Incremental Design/Development”. Sowohl die agile als auch die nutzerzentrierte Softwareentwicklung propagieren einen iterativen und inkrementellen Ansatz, da dadurch systematisch Feedback gesammelt und dieses schrittweise umgesetzt werden kann. Der iterative Ansatz erlaubt es hierbei, die Software im Erstellungsprozess basierend auf dem gesammelten Feedback formativ zu verfeinern. Im Gegensatz dazu verfolgt die inkrementelle Strategie eine schrittweise Gestaltung und Entwicklung der einzelnen Funktionalitäten. Dabei werden komplexe Softwareprodukte oftmals in einzelne Bestandteile wie beispielsweise die Schichten der Benutzerschnittstelle, der Geschäftslogik und der Datenbank zerlegt und sukzessive entwickelt. Zusammenfassend kann festgestellt werden, dass die iterative Erstellung kurze Feedbackzyklen ermöglicht und die inkrementelle

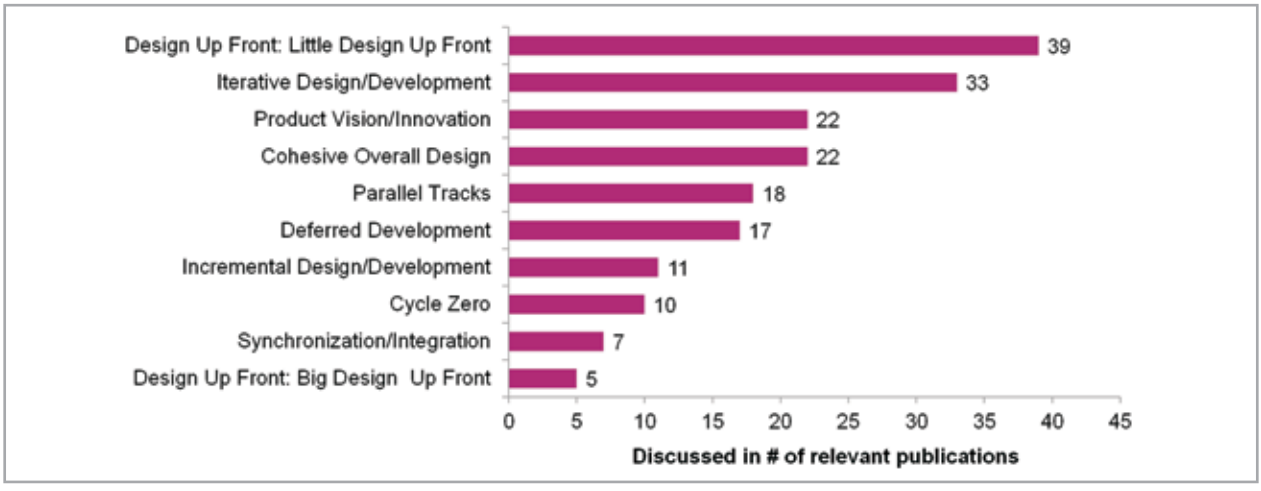

Abbildung 3: Kodierungen und Häufgkeiten in der Kategorie Prozesse

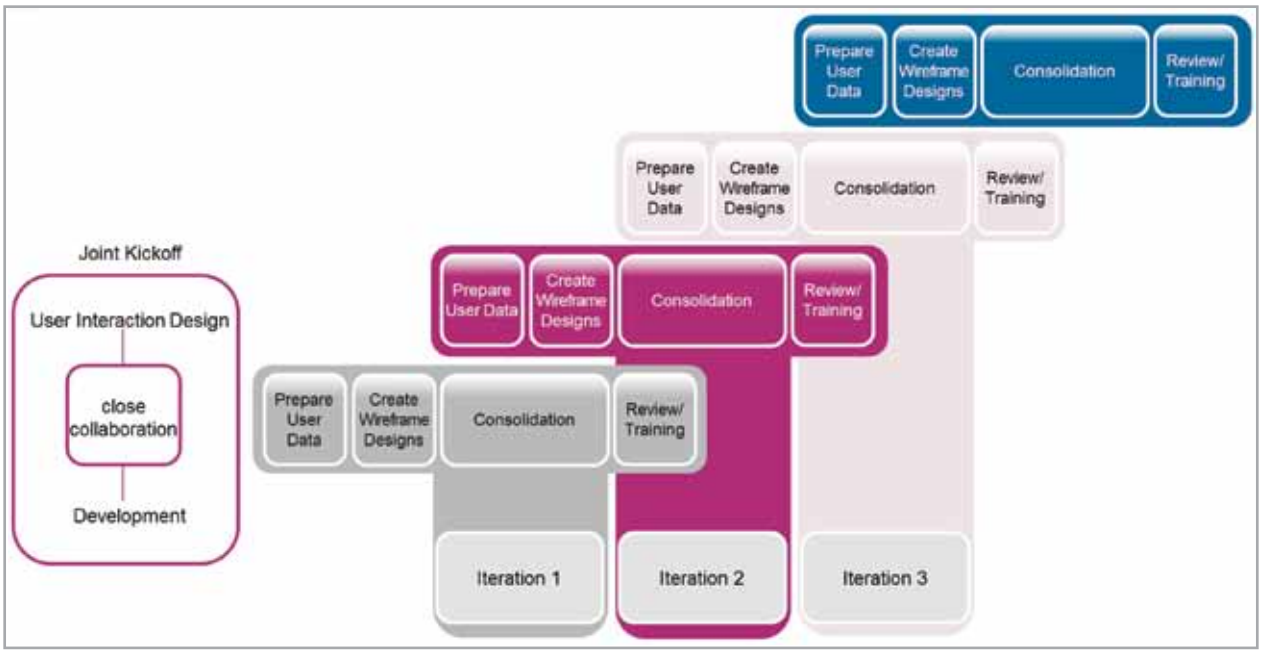

Abbildung 4: Parallele, synchronisierte Prozesse

Erstellung die Möglichkeit bietet, einzelne Bestandteile der Gesamtlösung vorab zu evaluieren.

Prinzip III: Parallele, Synchronisierte Prozesse

Das vorgestellte Prinzip I betont die Bedeutung einer Explorationsphase in der agilen und nutzerzentrierten Softwareentwicklung. In Ergänzung zu diesem Prinzip beschreibt das Prinzip III Ansätze zur prozessualen Ausgestaltung der Interaktion zwischen Gestaltungs- und Entwicklungsteam. Es fasst die drei Kategorien „Parallel Tracks“, „Design One Sprint Ahead“ und „Synchronization/Integration“ zusammen. Mit Prinzip III wird das Ziel beschrieben, Aktivitäten der nutzerzentrierten Gestaltung und der agilen Softwareentwicklung zu entkoppeln und zu parallelisieren. Definierte Synchronisationspunkte stellen hierbei eine Abstimmung und Integration der Aktivitäten sicher (vgl. Abbildung 4).

Prinzipien zu Praktiken

Neben den zuvor ausgeführten prozessorientierten Prinzipien wurden in der durchgeführten Literaturstudie die an Praktiken orientierten Prinzipien „Kontinuierliche Einbeziehung der Stakeholder“ und „Artefaktzentrierte Kommunikation“ identifiziert. Die Praktikenperspektive betrachtet im Gegensatz zur Prozessperspektive konkrete Methoden und Techniken, welche im Softwareentwicklungsprozess Anwendung finden können. Abstrakt lässt sich der Entwicklungsprozess von Software in die Phasen Analyse, Konzeptualisierung, Gestaltung/Entwicklung und Evaluation differenzieren. In den letzten Jahrzehnten wurden sowohl im Software Engineering als auch im Bereich der Human-Computer Interaction eine Vielzahl von Praktiken und Methoden entwickelt. Im Folgenden werden zwei zentrale Prinzipien vorgestellt, die basierend auf der analysierten Literatur eine Generalisierung der Praktiken und Methoden auf Konzeptebene beschreiben.

Prinzip IV: Kontinuierliche Einbeziehung der Stakeholder

Das Prinzip IV beschreibt den kontinuierlichen Einbezug von Stakeholdern in den Entwicklungsprozess. Aus Sicht der nutzerzentrierten Gestaltung werden Nutzer der zu erstellenden Software als primäre Stakeholder angesehen. Zusätzlich werden üblicherweise im Software Engineering auch Kunden (d.h. die Auftraggeber im Falle von Individualsoftware bzw. die Entscheider der Softwareauswahl im Falle von Produktsoftware) als Stakeholder betrachtet. Abbildung 5 stellt die aus der Literatur extrahierten Praktiken zur Einbeziehung der Stakeholder und die Häufigkeit von deren Nennung in den jeweiligen Publikationen dar. Das „Usability Testing“ stellt hierbei die am häufigsten genannte

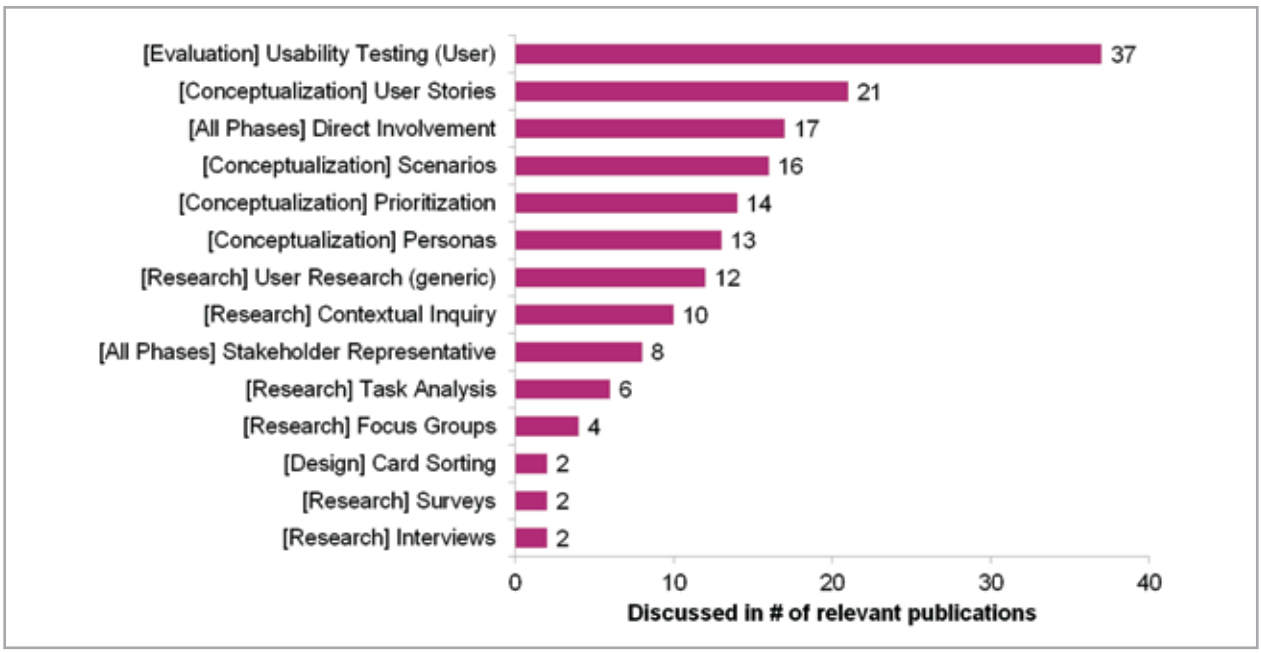

Abbildung 5: Praktiken zur kontinuierlichen Einbeziehung der Stakeholder

Praktik dar, gefolgt von der direkten Einbeziehung als generischer Methode sowie typische Aktivitäten in der Konzeptualisierungsphase wie beispielsweise „User Stories“ sowie „Szenarien“.

Prinzip V: Artefaktzentrierte Kommunikation

Das Prinzip V fokussiert auf die artefaktzentrierte Kommunikation. Sowohl die agile Softwareentwicklung als auch die nutzerzentrierte Gestaltung betonen die Bedeutung von Artefakten im Entwicklungsprozess. So forciert beispielsweise das agile Softwareentwicklungsmodell SCRUM lauffähige Software zum Ende eines jeden Entwicklungszyklus. Artefakte müssen jedoch nicht notwendigerweise lauffähige Software darstellen, es lassen sich vielmehr eine Vielzahl von alternativen Ansätzen zur Erzeugung von Artefakten im Entwicklungsprozess feststellen.

Sowohl Ergebnisse der Anwendung von Praktiken im Kontext der kontinuierlichen Einbeziehung von Stakeholdern wie beispielsweise konkrete „User Stories“ und „Szenarien“ als auch unterschiedliche Formen von Prototypen (siehe Abbildung 6) können den Kommunikationsprozess zwischen den am Entwicklungsprozess beteiligten Akteuren sowie den Stakeholdern unterstützen. Abbildung 7 stellt die identifizierten Praktiken und die Häufigkeit von deren Nennung in den Publikationen dar.

Zusammenfassung

Basierend auf einer interdisziplinären Perspektive wurde eine Literaturstudie zur agilen und nutzerzentrierten Softwareentwicklung durchgeführt. Die Literaturstudie verfolgte das Ziel, Wissen aus unterschiedlichen Disziplinen zu integrieren und in grundlegenden Prinzipien zu verdichten. Es konnten drei generalisierende Prinzipien zu Prozessen sowie zwei Prinzipien zu Praktiken identifiziert werden. Bei Betrachtung der identifizierten Prinzipien wird offensichtlich, dass agile Softwareentwicklung und nutzerzentrierte Gestaltung nicht im Widerspruch zueinander stehen, sondern vielmehr als komplementär betrachtet werden können.

Die vorgestellten Prinzipien können mittelständischen Softwareherstellern helfen, ihre aktuell eingesetzten Prozesse und Praktiken in der Softwareentwicklung zu reflektieren und gegebenenfalls an ausgewählten Stellen anzupassen beziehungsweise zu ergänzen. Hierbei wird ein schrittweises und bedarfsorientiertes Vorgehen empfohlen. Während die identifizierten Prozessprinzipien eher übergreifenden Charakter haben, können spezifische Praktiken zur Verfolgung einer kontinuierlichen Einbeziehung von Stakeholdern sowie einer artefaktzentrierten Kommunikation abhängig von dem jeweils konkret vorliegenden Szenario ausgewählt und angewendet werden.

Abbildung 6: Scribbles und interaktive Prototypen als Artefakte

Es ist wichtig zu betonen, dass die vorgestellten Prinzipien keinen Anspruch auf Vollständigkeit erheben. Insbesondere auf der Team- bzw. der organisationalen Ebene von Softwareherstellern liefern die aus der Literatur identifizierten Prinzipien keinen Beitrag. Im Rahmen der Mittelstand-Digital Förderinitiative „Usability“ werden die bisher identifizierten Prinzipien im Kompetenzzentrum „Usability in Germany“ (http://usability-in-germany.de) evaluiert und erweitert. Sowohl Softwarehersteller als auch Anwenderunternehmen können auf dieser Grundlage in der Entwickung, Auswahl, Einführung und Nutzung von Software unterstützt werden. Es ist geplant, die Prinzipien in ein Managementkonzept für den deutschen Mittelstand einfließen zu lassen. Über ein Netzwerk von Herstellern, Dienstleistern und Anwendern kann dann eine Diffusion des Konzeptes in den deutschen Mittelstand realisiert werden.

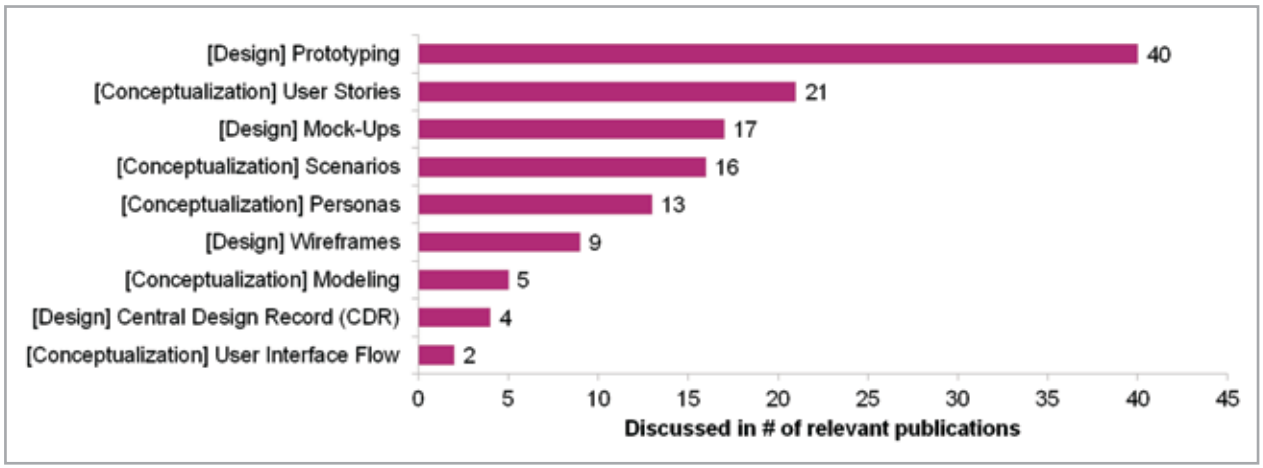

Abbildung 7: Praktiken zur Artefaktzentrierung

Literatur

Barksdale, J.T. & McCrickard, D.S. (2012). Software product innovation in agile usability teams: an analytical framework of social capital, network governance, and usability knowledge management, International Journal of Agile and Extreme Software Development, 1, 52–77.

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. Information and Software Technology, 50(9-10), 833–859.

Norman, D. (1988). The Design of Everyday Things Currency Doubleday. New York.

Kitchenham, B.A. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering: EBSE Technical Report EBSE-2007-01. Keele University, Keele, UK.

Autoren

Prof. Dr. Alexander Mädche ist Inhaber des Lehrstuhls für Wirtschaftsinformatik IV, Fakultät für Betriebswirtschaftslehre, Universität Mannheim sowie geschäftsführender Direktor des Instituts für Enterprise Systems (InES) an der Universität Mannheim.

Zuvor war er Abteilungsleiter der Forschungsgruppe „Wissensintensive Systeme“ am Forschungszentrum Informatik (FZI), Karlsruhe, Manager beim Business Intelligence Kompetenzzentrum der Bosch Gruppe, Stuttgart sowie Vice President Produktmanagement bei der SAP AG, Walldorf.

Seine Forschungsschwerpunkte sind die Entwicklung von Unternehmenssoftware (Softwareproduktentwicklung, Nutzerzentriertes Design, Web-Plattformen) sowie die Einführung und Nutzung von menschzentrierten, betrieblichen Informationssystemen.

Prof. Dr. Dieter Wallach ist Professor für Human-Computer Interaction und Usability Engineering im Fachbereich Informatik und Mikrosystemtechnik an der Fachhochschule Kaiserslautern.

Nach dem Studium der Psychologie, Informatik und Informationswissenschaft promovierte er im Graduiertenkolleg Kognitionswissenschaft an der Universität des Saarlandes bevor er als Postdoctoral Associate an der Carnegie Mellon University in Pittsburgh (PA) und der Universität Basel arbeitete. Er erhielt Rufe auf Professuren für Software Engineering, Neue Medien und Psychologische Ergonomie. Dieter Wallach lehrt und forscht seit 2001 in Kaiserslautern im Bereich User Interface Design und Usability Engineering und ist seit 2000 geschäftsführender Gesellschaft der Ergosign GmbH.