Michael Burmester, Joachim Machate und Nina Sandweg

Integration benutzerzentrierter Methoden in die Software-Entwicklung

Integrating User-Centered Design Methods with Software Engineering

Usability-Engineering_Software-Engineering_Entwicklungsprozess_benutzerzentrierte Gestaltung_Rational Unified Process (RUP)

Zusammenfassung. Der Ablauf vieler Software-Projekte orientiert sich oftmals an eingeführten Software-Engineering-Prozessen. Bei der Entwicklung interaktiver Produkte gewinnt Usability-Engineering zunehmend an Bedeutung. Somit entsteht die Notwendigkeit, dass Software-Engineering- und Usability-Engineering-Prozesse integriert werden. Der vorliegende Artikel zeigt Möglichkeiten auf, wie am Beispiel des Rational Unified Process (RUP) beide Prozesse pragmatisch integriert werden können.

Summary. The project cycle of numerous software projects is often based on the software engineering processes involved. When it comes to developing interactive products, usability engineering is increasingly gaining in importance. Hence, the necessity arises that software engineering and usability engineering processes have to be integrated.
Based on the "Rational Unified Process (RUP)", a well-known software engineering process, this article reveals possibilities to integrate both processes – usability and software engineering – in a pragmatic way.

1. Integration von Usability-Engineering-Maßnahmen in Software-Engineering-Prozesse

Bei der Entwicklung IT-basierter Produkte hat Usability als Qualitätskriterium einen entscheidenden Anteil am Erfolg eines Produktes. Die Erkenntnis, dass entsprechende Maßnahmen zur Sicherung dieser Qualität notwendig sind, setzt sich mehr und mehr in Gestaltungs- und Entwicklungsprojekten durch. Der Einsatz ingenieurwissenschaftlicher Vorgehensweisen und Methoden bei der Softwareentwicklung ist als Software-Engineering bekannt. In Analogie dazu wird das systematische Anstreben von Usability als Usability-Engineering bezeichnet.

Als wichtiges Hindernis für die Planung und den Einsatz von Usability-Engineering-Maßnahmen rückt dabei häufig in den Vordergrund, dass Software-Engineering-Prozesse unabhängig von Usability-Engineering-Maßnahmen geplant und durchlaufen werden. Dies führt zu Situationen, bei denen Zeit, Budgets und Ressourcen für Usability-Engineering-Maßnahmen oder -Prozesse im Rahmen der Softwareentwicklung nicht oder nur unzureichend vorgesehen werden. Usability-Engineering-Prozesse laufen oft losgelöst von Software-Engineering-Prozessen, was zu mangelnder Integration ihrer Arbeitsergebnisse führt.

Sehr häufig wird Usability-Engineering als reine qualitätsprüfende oder qualitätssichernde Maßnahme gesehen, bei der Arbeitsergebnisse, wie z.B. Benutzungsoberflächenprototypen, expertenbasiert oder empirisch überprüft werden. Da die Prüfung sehr häufig zu einem späten Zeitpunkt der Entwicklung stattfindet, können Ergebnisse, die starke Code-Änderungen zur Folge hätten, nicht mehr berücksichtigt werden. Usability-Engineering bietet aber weit mehr als Qualitätsprüfung am Ende einer Produktentwicklung. Den größten Nutzen aus dem Einsatz von Usability-Engineering-Methoden ziehen Projekte, die konsequent auf deren Einsatz bereits in der Anforderungserhebung und während des Systementwurfs setzen. Obwohl das Durchlaufen einer Requirementsphase zum Standardablauf eines jeden Softwareentwicklungsprojekts zählt, bleibt eine fundierte Auseinandersetzung mit den Anforderungen der Produktnutzer oftmals außen vor. Das erfolgreiche Einbinden von Benutzern in die Requirementsphase, so wie es die DIN EN ISO 13407 (2000) fordert, ist in der praktischen Anwendung vieler Software-Engineering Prozesse noch nicht konsequent genug etabliert (Gulliksen, Göransson, Lif 2001).

Ungeachtet der Tatsache, dass es bereits erfolgreiche Integrationen von Software-Engineering und Usability-Engineering-Prozessen gibt (IBM 1996; Geissbühler, Hassenzahl und Beu 2003), finden

sich in der Praxis immer wieder Unsicherheiten und manche Schwierigkeiten, aber auch Wege konstruktiv die Integration zu ermöglichen.

Aus der Perspektive von Usability-Fachleuten bestehen Unsicherheiten darin, wie Maßnahmen des Usability-Engineering in Software-Engineering-Prozesse integriert werden können, so dass Ergebnisse dieser Aktivitäten entsprechend genutzt werden. Meist sind Entwicklungsprozesse in den Unternehmen bereits etabliert und Usability-Engineering muss nachträglich integriert werden. Aus Sicht der Personen, die für die Planung und den Einsatz von Softwareentwicklungsprozessen verantwortlich sind, herrscht häufig Unsicherheit darüber, in welcher Phase eines Prozesses welche Maßnahmen durch wie qualifizierte Personen mit welchem zu erwartenden Ergebnis integriert werden müssen. Auf beiden Seiten gibt es zudem Verwirrung über ähnlich lautende Begriffe wie „Iteration" oder „Phase", die häufig aber unterschiedlich verstanden werden.

Im Folgenden soll aufgezeigt werden, welche Unterschiede zwischen Software-Engineering-Prozessen und Usability-Engineering-Prozessen bestehen und wie sich Usability-Engineering-Maßnahmen in einen bestehenden Software-Engineering-Prozess integrieren lassen. Exemplarisch wird dies am Rational Unified Process (RUP) als repräsentativer und häufig genutzter Software-Entwicklungsprozess verdeutlicht.

2. RUP als repräsentativer und häufig genutzter Softwa re-Entwicklungsprozess

Der Rational Unified Process® (RUP®) ist ein kommerzielles Produkt, das von Rational®Software entwickelt und vertrieben wurde und nun zu IBM (RUP IBM 2004) gehört. Der Prozess wird mittels einer eigenen Plattform unterstützt und ist auf breiter Basis als Quasi-Standard angesehen (Gulliksen, Göransson, Lif 2001).

Die folgende Darstellung des RUP lehnt sich an die Beschreibung von Kruchten (2000) an. Charakteristisch für den RUP ist seine auf den konkreten **Use Case** bzw. Anwendungsfall gerichtete

Bild 1: Iteration im RUP (nach Kruchten, 2000)

Sicht der Softwareentwicklung, die eine eindeutige Beschreibung der Anforderungen vorsieht und den gesamten Entwicklungsprozess einrahmt. Der RUP ist **architekturzentriert**. Während das Use Case Model die Funktionalität eines Systems beschreibt, unterstützen die architektonischen Sichten des RUP (Logik, Implementierung, Prozesse, Verteilung und übergreifend die Use Case Sicht) den Entwurf und die Implementierung des Systems. Seine **iterativ-inkrementelle Vorgehensweise** verbessert Schritt für Schritt die Qualität des Produktes. Eine Iteration ist ein „Mini-Projekt", das als Ergebnis eine Version des Produkts hat, die intern oder extern ausgeliefert wird. Jede Version soll gegenüber ihrer Vorgängerversion eine inkrementelle Verbesserung (bzgl. Funktionalität und Qualität) aufweisen (vgl. Bild 1). Das Ergebnis einer Iteration wird Inkrement genannt.

Der Prozess wird in zwei Dimensionen beschrieben. Dies sind zum einen die zeitliche (dynamische) Dimension und zum anderen die statische (inhaltliche) Dimension.

Die zeitliche Dimension gliedert sich in vier **Phasen**.

- Inception Phase - Konzeptualisierung
  Im Rahmen einer Machbarkeitsanalyse werden wesentliche Geschäftsvorfälle spezifiziert, die angestrebte Funktionalität skizziert und Kosten und Risiken ermittelt. Es werden alle Akteure identifiziert, mit denen das System in Interaktion tritt und die relevanten Use Cases identifiziert und teilweise beschrieben. Zudem werden für das Projekt Erfolgskriterien, Risikoeinschätzungen und Aufwandabschätzungen erstellt.
- Elaboration Phase – Entwurf
  In dieser Phase wird das System analysiert, Funktionalität festgelegt und die Systemarchitektur entworfen. Die Use Cases werden zu mindestens 80% fertig gestellt. Endprodukt dieser Phase ist ein technisches System, das als Grundlage für die weitere Implementierung dient und in dem die wesentlichen Anforderungen umgesetzt worden sind.
- Construction Phase – Konstruktion
  Das Produkt wird in dieser Phase bis zur Auslieferungsreife entwickelt. Für die Konstruktionsphase ist charakteristisch, dass keine gravierende Änderung der in den beiden vorangegangenen Phasen ermittelten Anforderungen mehr erfolgen, sondern vielmehr das System Zug um Zug umgesetzt wird.
- Transition Phase – Übergang
  Zentrale Aufgabe der Phase Übergang ist die Bereitstellung und Integration des Produkts in dessen zukünftigem Einsatzort. Enthalten sind Aktivitäten wie Auslieferung, Training, Einsatzunterstützung und Wartung.

Die inhaltliche Dimension umfasst die statischen Aspekte des Prozesses. Dazu gehören unterschiedliche Rollen (Workers), Arbeitsergebnisse (Artifacts), Aufgaben (Activities) und Disziplinen. Mit den Rollen werden die Verantwortlichkeiten für bestimmte Aufgaben im Prozess

Tabelle 1: Prozess-Disziplinen im RUP

Prozessdisziplinen	Aktivitäten
Geschäftsprozess-modellierung	Betrachtung und Dokumentation des Geschäftsmodells, für das das System entworfen werden soll
Anforderungsmanagement	Anforderungsanalyse, Dokumentation und Organisation
Analyse & Design	Design der Architektur auf Basis der erhobenen Anforderungen
Implementierung	Implementierung des Systems
Test	Systemverifikation
Verteilung	mit der Inbetriebnahme des Systems zusammenhängenden Aktivitäten wie beispielsweise Installation und Schulung

festgelegt. Die Aufgaben haben klar definierte Ziele und liefern ein Ergebnis: das Artefakt. Die **Disziplinen** existieren phasenübergreifend. Sie sind Kernprozesse, die sich aus einer Gruppierung logisch miteinander verknüpfter Aktivitäten zusammensetzen. Es werden sechs primäre Prozessdisziplinen (vgl. Tabelle 1) sowie drei Infrastrukturdisziplinen unterschieden (vgl. Tabelle 2).

Obwohl alle Disziplinen des RUP ihren Beitrag zu jeder Iteration leisten (vgl. Bild 1), wird nicht jeder Disziplin der gleiche zeitliche und arbeitsintensive Aufwand gewidmet. Beim Anforderungsmanagement lässt sich beispielsweise erkennen, dass in frühen Iterationen und Phasen viel Arbeit in diese Disziplin einfließt, während in späteren Phasen der Entwicklung nur noch geringfügige Änderungen anfallen. Gleiches gilt für die Artefakte, die reifen und sich vervollständigen (vgl. Bild 2). Dabei wird Usability-Engineering nicht als eigene Disziplin des RUP angesehen, aber entsprechende Aspekte und Methoden finden sich in verschiedenen Disziplinen und Aktivitäten, wie Kapitel 5 zeigen wird.

Tabelle 2: Disziplinen der Infrastruktur im RUP

Infrastrukturdisziplinen	Aktivitäten
Konfigurations- und Änderungsmanagement	Konsistente Ergebnisablage und kontrollierte Änderungsberücksichtigung
Projektmanagement	Projektplanung und –steuerung
Umgebung	Projektunterstützende Verfahren und Werkzeuge

3. Benutzerzentrierte Gestaltung als zentraler Ansatz

Nach DIN EN ISO 9241-11 (1998) wird unter Gebrauchstauglichkeit bzw. im Englischen „Usability" eine Qualität der Nutzung verstanden. Usability lässt sich nicht als eine Produkteigenschaft festmachen, sondern kann nur im Zusammenspiel zwischen Nutzereigenschaften, Aufgaben bzw. Ziele der Benutzer, der organisatorischen und sozialen sowie der technischen und physischen Umgebung bestimmt werden. Mit anderen Worten: das Produkt muss sich in einen bestimmten Nutzungskontext einpassen, und es muss Benutzern erlauben, ihre Ziele effektiv, effizient und zufrieden stellend zu erreichen (DIN EN ISO 9241-11, 1998).

Wie in der Produktentwicklung Usability erreicht werden kann, beschreiben verschiedene Autoren als Benutzerzentrierte Gestaltung, z.B. Gould und Lewis (1985), Hix und Hartson (1993), Nielsen (1993), Beyer und Holtzblatt (1998), Mayhew (1999), Rosson und Carroll (2002), sowie „user centered design" bzw. „Benutzer-orientierte Gestaltung interaktiver Systeme" der DIN EN ISO 13407 (2000). In ihrem Verständnis eines Usability-Engineering-Prozesses findet sich eine breite Übereinstimmung zentraler Kriterien:

- Benutzerbeteiligung: Die Benutzer und andere Interessensvertreter werden systematisch mit in den Gestaltungsprozess eingebunden.
- Multidisziplinäres Team: Um unterschiedliche Perspektiven bei der Gestaltung eines Produktes zu berücksichtigen, wird empfohlen Experten aus verschiedene Disziplinen am Gestaltungsprozess zu beteiligen. Dazu gehören u.a. Produktmanager, Projektmanager, Software-Entwickler, Graphikdesigner, Usability-Fachleute, Linguisten, sowie Vertreter aus Marketing und Vertrieb.
- Prozessphasen: Die Prozesse lassen sich im Wesentlichen als vier Phasen oder Aktivitäten darstellen: 1. Verstehen des Nutzungskontextes und der daraus resultierenden Anforderungen an das zu gestaltende Produkt und Festlegung von Usability-Zielen (Nutzungskontextanalyse, „Analyse" in Bild 3), 2. Entwerfen der Funktionen und der Benutzungsschnittstelle (Entwurf und Gestaltung, „Gestaltung" in Bild 3), 3. Erfahrbar machen ausgewählter Funktionen und der Benutzungsoberfläche (Prototyping in Bild 3) und 4. Prüfen auf Einhaltung der Usability-Ziele und Ermitteln von Optimierungspotenzialen („Evaluation" in Bild 3).
- Datenerhebung: Bei der Entwicklung von Benutzungsoberflächen lässt sich eine Vielzahl von Informationsquellen nutzen. Als unverzichtbar gelten hier empirische Methoden wie Befragungen von Benutzern, Messungen an Arbeitsplätzen (z.B. Häufigkeiten und Bearbeitungszeiten von Aufgaben, sowie Gewichtungen von Aufgaben anhand von Fragebögen), oder Beobachtungen der Arbeitstätigkeit am Arbeitsplatz, etc. Unter einer analytischen Perspektive können bestehende Dokumente oder Produkte im Detail untersucht werden. Hierbei bleibt zwangsläufig ungesichert, ob die Analysen oder abgeleiteten Ideen und Gestaltungsentscheidungen den tatsächlichen, realen Verhältnissen entsprechen, da lediglich Annahmen über die Benutzer und ihre Bedürfnisse getroffen werden.
- Methoden: Jede einzelne Phase des benutzerzentrierten Gestaltungsprozesses verfolgt ihr eigenes phasenspe-

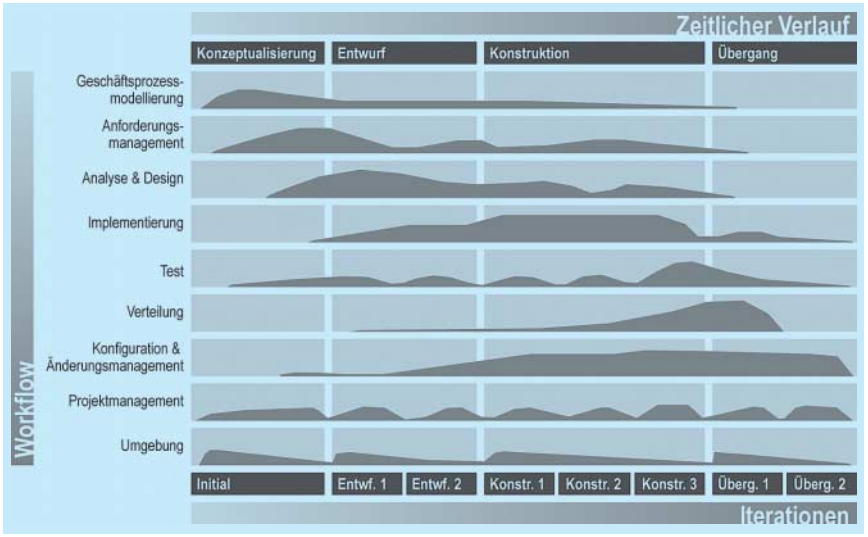

Bild 2: Hügeldiagramm zur Veranschaulichung der zeitlichen Verteilung von Aktivitäten der verschiedenen Disziplinen (nach Kruchten, 2000)

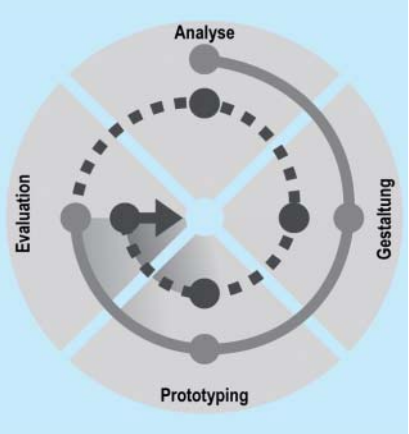

Bild 3: Iterativer Prozess im Usability-Engineering

zifisches Ziel auf dem Weg zum optimalen Produkt. Bewährte Methoden sollen dabei sicherstellen, dass die Ziele mit einiger Zuverlässigkeit erreicht werden können.

- Iteration: Der Benutzerzentrierte Gestaltungsprozess wird häufig als iterativer Prozess mit den oben genannten Phasen beschrieben. Die Ergebnisse der Evaluation werden zum einen verwendet, um die Beschreibung des Nutzungskontexts zu präzisieren. Zum anderen fließen Evaluationsergebnisse direkt in die Gestaltung und das Prototyping zurück. Bei jeder Evaluation wird die Erreichung der Usability-Ziele überprüft und entschieden, ob eine weitere Iteration stattfinden muss.

„Benutzerzentriert" bedeutet mehr, als dass die Belange potenzieller Benutzer lediglich berücksichtigt werden. „Benutzerzentriertes" Gestalten unterstreicht, dass neben der Analyse von Benutzereigenschaften, Aufgabenmerkmalen und Umgebungsfaktoren die direkte Benutzerbeteiligung am Gestaltungsprozess von zentraler Bedeutung ist. D.h. potenzielle Benutzer sollen möglichst häufig neue Entwürfe bewerten, aber auch aktiv als fachliche Experten an Ideenfindung und Design teilnehmen.

4. Konzeptuelle Unterschiede

4.1 Nutzungskontextanalyse

Anforderungen im Usability-Engineering und RUP

Die Ansicht, was eigentlich Anforderungen sind und wie sie formuliert werden, variiert je nach Entwicklungsmethodik und Prozesssicht. Zunächst sollen die Sichtweisen aus Usability-Engineering-Prozessmodellen dargestellt werden. So vertreten Beyer & Holtzblatt die Meinung, dass Anforderungen nicht dazu geeignet seien, um Design zu kommunizieren, sondern sie sollen exakt beschreiben, was das zukünftige System ausmacht (Beyer & Holtzblatt, 1998, S. 369). Anforderungen werden als eine Liste von Funktionen sowie nicht-funktionalen Eigenschaften, z.B. Sicherheit, Performance, etc. definiert. Eine ähnliche Sicht vertreten Rosson & Carroll, die ebenfalls Anforderungsspezifikation als ein Dokument beschreiben, das die gewünschten Eigenschaften eines zukünftigen Systems beinhaltet (Rosson & Carroll, S. 38). Deborah J. Mayhew beschreibt Anforderungsspezifikation innerhalb des Usability-Engineering Life Cycle als eine Sammlung von Benutzerprofilen, Aufgabenanalysen, Zielplattformeigenschaften und allgemeinen Entwurfsprinzipien (Mayhew 1999, S. 34). Dabei sieht Mayhew eine Wechselwirkung zwischen Usability- und Software-Engineering. Beispielsweise können Ergebnisse einer nutzungskontextbezogenen Aufgabenanalyse als Ausgangspunkt für die Identifikation der im Softwareentwicklungsprozess benötigten Use Cases dienen. Umgekehrt wiederum können Use Cases als Ausgangspunkt für eine Nutzungskontextanalyse verwendet werden. Eine Verschmelzung durchaus ähnlicher Aktivitäten findet leider nicht statt, sondern beide Prozesse laufen parallel nebeneinander oder überlappen sich teilweise.

Anders als das in den Usability-Engineering-Prozessen formulierte Verständnis einer Anforderungserhebung ist im RUP eine breitere Sicht des Anforderungserhebungsprozesses verankert. Der RUP Requirements-Workflow beginnt seine Aktivitäten während der Anfangsphase, der Inception Phase. An deren Ende steht eine Beschreibung der Stakeholder (von der Produktentwicklung betroffene Personen) sowie ein klares Verständnis der Anforderungen, die sich aus den Geschäftsprozessen und High Level Business Use Cases ergeben. Die anschließende Ausarbeitungsphase (Elaboration Phase) liefert eine umfassende Anforderungsdokumentation, die neben einem vollständigen Use Case Model (Beschreibung der Aktoren plus Use Cases) auch zusätzliche, nicht-funktionale Anforderungen enthält. Das FURPS-Modell des RUP unterteilt Anforderungen nach funktionalen Anforderungen, (F)unctionality und nicht-funktionalen Anforderungen, die nach (U)sability, (R)eliability, (P)erformance und (S)upportability Aspekten unterschieden werden. In den meisten Projekten werden allerdings die funktionalen Anforderungen den Großteil der Anforderungen ausmachen, die übrigen sich auf einen kleineren Rest verteilen. Vielen Lesern dürfte der Dilbert-Cartoon bekannt sein, in dem ein Projektleiter gefragt wird, ob er sich bewusst ist, dass bei der von ihm aufgelisteten Menge an Features niemand in der Lage sein wird, ein derart komplexes System zu benutzen. Seine Antwort darauf lautet: „Guter Einwand, wir sollten besser leichte Bedienbarkeit zu den Features hinzufügen."

Use Cases und Aufgabenanalyse

Neben zentralen Eigenschaften des RUP wie Anpassbarkeit des Prozesses und Unterstützung durch geeignete Software-Tools gilt als dritte herausragende Eigenschaft, dass der RUP einen Use Case getriebenen Ansatz verfolgt (Kruchten 2000, S. 30; Kruchten, Ahlquist, Bylund 2001, S. 161). Dies bedeutet, dass Use Cases die Grundlage für den gesamten Entwicklungsprozess bilden. Im Verständnis des RUP sind Use Cases stark systemzentriert, sie beschreiben eine Reihenfolge von Aktionen, an deren Ende etwas Sinnvolles für einen bestimmten Aktor dabei herauskommen soll, und dienen vorrangig der Beschreibung und Umsetzung von Systemfunktionen mittels einzelner Teilschritte. Der ursprünglich von Kruchten (2000, S. 101) gewählte Stil mit rein auf das System bezogenen Aktionen (‚Das System tut dies…') wurde mittlerweile in vielen Fällen zugunsten eines stärker an einem wechselseitigen Dialog zwischen Akteuren ausgerichteten Stil geändert. Eine erweiterte Sicht auf Use Cases, die statt auf eine Beschreibung der Systemfunktionalität auf die Ziele der Benutzer ausgerichtet, aber durchaus kompatibel mit dem Verständnis des RUP ist, vertritt Alistair Cockburn (Cockburn 2001). Welche äußerliche Form auch immer für die Beschreibung von Anforderungen mittels Use Cases gewählt wird, hängt von den jeweiligen Projektbedürfnissen und Rahmenbedingungen ab. Während die Beschreibung eines zukünf-

tigen Systems im RUP sich primär an der intendierten Funktionalität ausrichtet, stehen beim Usability-Engineering zukünftige Systemnutzer im Fokus des Interesses. Zur Analyse und Beschreibung des zukünftigen Nutzer-Systemverhaltens und zur Formulierung der daraus resultierenden Anforderungen werden in Usability-Engineering-Prozessen in der Regel Aufgabenanalyseverfahren eingesetzt. Durchaus ähnlich im Vergleich zum Anforderungsmanagement des RUP ist, dass die Aufgabenanalyse im Usability-Engineering alle Phasen des Entwicklungsprozesses mit jeweils unterschiedlichen Methoden und Schwerpunkten überspannt (Redish & Wixon 2003, S. 926). Neben den Zielen der Benutzer spielen Nutzungsumgebung und Nutzerprofile eine wesentliche Rolle im Verständnis des Nutzungskontextes.

Zwischen Aufgabenanalyse und Use Case Modelling gibt es somit wichtige Unterschiede:

- Mit der Aufgabenanalyse wird das Verhalten der Nutzer abgebildet. Externe Aufgaben spiegeln die Aufgaben ohne technologische Unterstützung wider. Handlungsabläufe und -strukturen sind bei internen Aufgaben bereits durch in der Nutzung befindlicher Werkzeuge beeinflusst. Der Fokus der Untersuchung, Analyse und Modellierung liegt deutlich darauf was der Nutzer macht. Gulliksen, Göransson und Lif bemerken dazu „there is [im RUP] little support for the analysis of the user tasks" (2001, S. 294).
- Im Fokus der Use Cases steht dagegen eher das Verhalten des Systems mit einem Aktor, der nicht notwendigerweise ein Nutzer sein muss.

Anforderungen und Design

Requirements-Engineering nach RUP beinhaltet nicht nur die Formulierung von Anforderungen, sondern bereits auch die Beschreibung und Prototypen der Benutzungsoberfläche. Letztere ist im Usability-Engineering explizit in eine andere Arbeitsphase - die Gestaltung - verlegt worden. Die Gestaltung der Benutzungsschnittstelle hat im Benutzerzentrierten Gestaltungsprozess eine eigene zentrale Bedeutung, denn „so weit es den Verbraucher betrifft, ist die Oberfläche das eigentliche Produkt" (Raskin, 1999, S. 22).

Weg und Ziel

Werden Arbeiten betrachtet, bei denen es um die Einbindung von Usability-Aspekten in RUP geht, z.B. Kruchten, Ahlquist, Bylund 2001; Phillips, Kemp 2002, dann lässt sich feststellen, dass die Modellierung sehr stark im Vordergrund steht. Es geht nicht so sehr um die Methoden, mit denen beispielsweise Anforderungen erhoben, Interaktionen entworfen oder Feedback zu einem Prototypen ermittelt werden. Vielmehr geht es um die strukturierte Modellierung der Ergebnisse und weniger, wie die Ausgangsdaten für eine Modellierung (z.B. für einen Use Case) erhoben werden. Im Gegensatz dazu stehen beim Benutzerzentrierten Gestaltungsprozess eher die Methoden zum Generieren der jeweils notwendigen Informationen im Vordergrund. Dies wird im nächsten Abschnitt zu den Quellen der Anforderungen noch einmal besonders deutlich.

Quellen der Anforderungen

Ein häufig genanntes Argument gegen das Wasserfall-Modell oder vergleichbare Softwareentwicklungsprozesse ist, dass Anwender und im weiteren Sinn Stakeholder nicht genau sagen können, welche Anforderungen sie eigentlich haben. Erst dann, wenn sie die Möglichkeit haben, ein System zu erleben, wird ihnen klar, welche Funktionalität sie vermissen oder für überflüssig halten. Nicht selten ist dieses Verhalten Ursache für Befürchtungen seitens des Projektteams, dass mit jedem neuen Prototyp neue Anforderungen entstehen und dies sozusagen zu einem nie endenden Fluss von Iterationen führt. Von daher nimmt die Kommunikation von Anforderungen zwischen Entwicklungsteam und Nutzerteam eine zentrale Rolle im Entwicklungsprozess ein.

Kasten 1 soll verdeutlichen, welche Quellen neben bzw. anstatt der auf empirischen Datenerhebung basierenden Nutzungskontextanalyse herangezogen werden. All die in Kasten 1 genannten Quellen für eine Spezifikation der Anforderungen haben ihre Berechtigung und sollen nicht in Frage gestellt werden. Was dabei allerdings oftmals auf der Strecke bleibt, ist eine direkte Auseinandersetzung mit den Bedürfnissen der Benutzer (Gulliksen, Göransson, Lif 2001, S. 294). Sofern dies überhaupt geschieht, wird dabei gern auf die Erfahrungen der Hotline und des Kunden-Supports zurückgegriffen, oder, falls vorhanden, die FAQ-Sektion der Produkt-Webseite zu Rate gezogen. Bei vielen großen Dienstleistern bestimmen Benutzervertreter oder Fachausschüsse als Mittler zwischen Projektteam und Produktnutzer die Bedürfnisse der Benutzer, selbst dann, wenn sie selber schon seit Jahren keine aktive Erfahrung mehr im Umgang mit dem Produkt haben.

4.2 Iteration

Im RUP werden in einer Iteration die Disziplinen Geschäftsprozessmodellierung, Anforderungsmanagement, Analyse und Design, Implementierung und Test durchlaufen. Ziel ist es, mit einer Iteration ein lauffähiges Teilsystem zu entwickeln (vgl. Bild 1). Beim Usability-Engineering wird Iteration anders verstanden. Hier dient die Iteration zur Optimierung des Benutzungsoberflächenkonzeptes. In der Evaluationsphase wird zum einen ermittelt, ob die Usability-Ziele erreicht wurden und zum anderen welche Aspekte der Gestaltung optimiert, sowie welche Erkenntnisse zum Nutzungskontext und welche Anforderungen korrigiert bzw. erweitert werden müssen. Im Gegensatz zum RUP entstehen hier keine Teilsysteme, sondern Prototypen, mit denen bestimmte Ziele verfolgt werden, wie z.B. Verdeutlichung eines Arbeitsablaufes auf der Basis einer Slide-Show (horizontaler Prototyp) oder Verdeutlichung einer neuen Eingabetechnik auf der Basis einer kleinen interaktiven Software, die genau diese Interaktion in vollem Umfang repräsentiert.

4.3 Phasen

Die Phasen Konzeptualisierung, Entwurf, Konstruktion und Übergang des RUP (vgl. Bild 2) spiegeln den zeitlichen Ablauf eines gesamten Softwareentwicklungsprojektes wieder. Bei dem oben beschriebenen Benutzerzentrierten Gestaltungsprozess ist zwar auch ein zeitlicher Aspekt enthalten, dieser gilt aber nur für die Reihenfolge der verschiedenen Aktivitäten Analysieren, Gestalten, Prototyping und Evaluieren (vgl. Bild 3). In der Evaluation wird anhand des Erreichens bzw. nicht Erreichens der Usability-Ziele entschieden, ob nochmals ein Durchlauf der Aktivitäten stattfindet. Dieser zweite Durchlauf steht dann im Zeichen der Optimie-

Kasten 1: Anforderungen erheben – woher kommen die Anforderungen?

- Der Produktmanager „weiß" was seine Kunden brauchen.
  Der Projektmanager wendet sich an den Produktmanager und fragt nach dessen Sicht auf die Anforderungen der Benutzer. Als häufiger Beleg für die Realitätstreue wird dabei vorgebracht, dass der Produktmanager weiß, was seine Kunden brauchen. Dieses Wissen entspringt seiner jahrelangen Erfahrung beim Verkauf und der Diskussion mit seinen Kunden. Und trotzdem handelt es sich um Annahmen des Projektmanagers darüber, was die Nutzer benötigen, wie sie wirklich Arbeiten, was ihnen bei der Arbeit wirklich helfen würde. Diese Annahmen können falsch sein, weil sie nicht direkt im Nutzungskontext erhoben und nicht mit Benutzern geprüft wurden.
- Der Kunde ist häufig der Einkaufs-Entscheider, aber nicht der Nutzer.
  Der Kunde des Produktmanagers ist in der Regel nicht der Nutzer, sondern der Einkaufs-Entscheider. Dies führt zwar zu einer Auseinandersetzung mit der gewünschten Funktionalität und bestimmten Leistungsmerkmalen, aber wie der Produkt-Nutzer mit dem Produkt zurechtkommt und welche speziellen Anforderungen aus dem Umgang mit dem Produkt entstehen, bleibt vage.
- Aus Vorgängerprodukten werden Spezifikationen.
  Eine nahezu unerschöpfliche Quelle von Anforderungen bieten Vorgängerprodukte. Getreu dem Motto „Was da drin war, kann nicht falsch sein" werden Funktionalität und oftmals auch Interaktionsmuster ungeprüft auf das neue Produkt übertragen. Dies führt zu Systemen, die viele Funktionen aufweisen, die nur noch aus historischen Gründen vorhanden sind. Die Übernahme von Interaktionsmustern kann im Sinne der Erwartungskonformität sinnvoll sein. Zu prüfen ist jedoch, ob die Interaktionsformen im Vorgängerprodukt wirklich effektiv und effizient waren, oder ob die Nutzer möglicherweise einfach Work-Arounds um das mangelhafte Interaktionsmuster gebaut haben.
- Die Konkurrenz zeigt was en-vogue ist.
  Neben dem Verweis auf das Profil eines Vorgängerprodukts, spielen Konkurrenzprodukte eine wichtige Rolle. So genannte Me-Too Features finden damit Eingang in das Anforderungsheft. ABER: Auch die Konkurrenz kann irren. Es ist oft unklar, wie die Konkurrenz zu ihren Features gekommen ist und ob diese Features von den Nutzern tatsächlich gewünscht werden.
- Change Requests
  Gerade bei Produkten, die nicht als off-the-shelf Produkt ihren Weg direkt vom Handel auf die Festplatte finden, sondern durch den Hersteller vor ihrem endgültigen Einsatz auf den jeweiligen Einsatzort angepasst werden, findet sich in der Regel eine Fülle von Kundenmeldungen, die als Change Requests einfach als zusätzliches Feature hinzu implementiert werden. Dies wiederum führt zu unbegrenztem und ungezügeltem Feature-Wachstum, das die Entwicklung eines Nachfolge-Produkts erheblich beeinflusst. Häufig werden die Hintergründe von Change Requests nicht ergründet. Um sinnvolle Entscheidungen über Anforderungen zu treffen, ist es notwendig das dahinter stehende Bedürfnis zu verstehen. Nur so kann entschieden werden, ob die Anforderung der Mehrheit der Nutzer zu gute kommt. Auf der Basis des Verständnisses der Change Requests lassen sich diese dann auch priorisieren.
- Der Verband empfiehlt.
  Als weitere Quellen für die Definition von Anforderungen dienen Kaufentscheidungshilfen (Purchase Selection Guides), die von Autoritäten, wie z.B. Branchenverbänden, herausgegeben werden. Hier werden sicher sinnvolle Hinweise gegeben, zu prüfen ist jedoch, ob die dort genannten Anforderungen tatsächlich ausreichen, oder ob sie eine Minimalforderung repräsentieren.
- Ausschreibungen
  Schließlich bieten Ausschreibungen (Request for Proposal), die mit Hilfe von Checklisten potenzielle Anbieter befragen, einen zusätzlichen Fundus an Anforderungen, die, sofern sie nicht bereits im Produkt berücksichtigt sind, so ihren Weg in den Anforderungskatalog finden.

rung der Vorstellung vom Nutzungskontext, der Gestaltungsideen, der Prototypen und einer erneuten Bewertung des Benutzungsoberflächenkonzeptes. Die Phasen Analysieren, Gestalten, Prototyping und Evaluieren im Benutzerzentrierten Gestaltungsprozess können somit in dieser Reihenfolge mehrfach hintereinander durchlaufen werden.

4.4 Testen

Im RUP werden in der Testphase die Funktionen geprüft. Die Frage ist, ob die Funktionen so implementiert wurden, wie sie geplant und spezifiziert waren, und ob das System so funktioniert, wie es gewünscht ist. Testen beim Usability-Engineering hat andere Ziele. Mit formativer Evaluation sollen Nutzungsprobleme mit dem Ziel ermittelt werden, die Gestaltung der Benutzungsoberfläche zu verbessern. Hier ist der User Test eine der bekanntesten Evaluationsmethoden, die vor allem formativ angewandt wird (Hassenzahl & Burmester, 1999). Formative Evaluation findet aber im iterativen benutzerzentrierten Gestaltungsprozess auf der Basis von Prototypen statt. Ein bereits fertiges System auf Nutzungsprobleme hin zu untersuchen macht meist wenig Sinn; denn die entdeckten Nutzungsprobleme würden zu hohen Änderungskosten führen, da Code umgeschrieben werden muss, oder können erst in der nächsten Version des Produktes berücksichtigt werden. Somit sind Testen im Entwicklungsprozess und Usability-Testen trotz der Ähnlichkeit der Bezeichnungen etwas komplett Verschiedenes.

Zudem muss erwähnt werden, dass in der ursprünglichen Konzeption des RUP User Testing eher kritisch gesehen wird. Kruchten, Ahlquist und Bylund (2001, S. 183) pflichten zwar bei, dass Feedback zu Prototypen eingeholt werden soll „it is immensely important to expose the user interface prototype to others, but to get valuable feedback you don't have to go through full-blown use tests in which real users perform real tasks with a prototype". Zudem wird eine gewisse Skepsis gegenüber Nutzerbeteiligung ausgedrückt. Die Autoren schreiben „…another, rather overestimated, way of exposing the design is to perform use tests" (ebd. S. 184).

5. Usability-Engineering in SE-Prozesse integrieren – aber wie?

5.1 Usability-Engineering – so früh wie möglich

Eine zentrale und seit den Anfängen des Usability-Engineering beständige Forderung ist, dass Usability-Engineering-Maßnahmen so früh wie möglich im Software-Engineering-Prozess integriert werden müssen. Studien z.B. von Gilb (1988 zitiert nach Bias & Mayhew, 1994) und Karat (1997) haben gezeigt, dass der späte Einsatz von Usability-Maßnahmen enorme Kosten verursachen kann. Der Grund dafür liegt darin, dass die tatsächlichen Arbeitsweisen der Nutzer Anforderungen sowohl an die Funktionalität als auch an die Präsentation von Informationen und die Gestaltung der Dialoge stellen. John, Bass und Adams (2003) merken dazu an, „usability requirements have an impact on architecture design and […] architectural decisions can preclude delivery of a usable system". Wird dies nicht angemessen umgesetzt, so wird die Nutzung des Systems ineffektiv und ineffizient. Häufig führt dies dann zu mangelnder Akzeptanz und zu Produktivitätseinbußen und schließlich zu Nachbesserungsforderungen gegenüber den Softwareherstellern.

5.2 Geschäftsprozessmodellierung aus Sicht des RUP

Um sicherzustellen, dass das zu entwickelnde System (Produkt) dem Benutzer bzw. der Organisation, in der das System genutzt wird, einen Mehrwert bringt, muss vor der Erstellung der Software das geschäftliche Umfeld bekannt sein. Bei der Geschäftsprozessmodellierung wird die bestehende Organisation dargestellt und eine Vision entwickelt, wie künftige Geschäftsprozesse bearbeitet werden können.

Die wesentlichen RUP-Rollen innerhalb der Geschäftsprozessmodellierung sind der Geschäftsprozessanalytiker und der Geschäftsprozessdesigner. In Zusammenarbeit mit diversen Interessensvertretern aus der zu modellierenden Organisation erarbeitet der Geschäftsprozessanalytiker die wesentlichen Business Use Cases und die involvierten Beteiligten. Er beschreibt sie mit Hilfe eines Business Use Case Modells, das sich aus Akteuren und Business Use Cases zusammensetzt. Parallel dazu sammelt er wichtige Definitionen und Konzepte des Geschäftsprozesses. Der Geschäftsprozessdesigner beschreibt die Business Use Cases im Detail, identifiziert die einzelnen Rollen und deren Verantwortlichkeiten, sowie die Produkte des Prozesses. Diese Beschreibung legt zunächst die formelle Sicht der Organisation fest. Dies ist auch für die Nutzungskontextanalyse notwendig. So können relevante Benutzergruppen für Feldbeobachtungen oder Befragungen ausgewählt werden. Zudem ergeben die formellen Workflows und Aufgaben ein Rahmenwerk für die Nutzungskontextanalyse. Dieses Rahmenwerk gilt es in der praktischen Umsetzung zu untersuchen. Zu beachten ist, dass Systeme die reale Arbeit der Nutzer unterstützen müssen und nicht nur die formelle Sicht der Organisation. In der Praxis unterscheiden sich Arbeitsabläufe, wie sie durch einen Arbeits- oder Organisationsgestalter vorgedacht und definiert wurden, von den tatsächlichen Arbeitsabläufen, wie sie in einer Organisation gelebt werden. Wie jenseits der formalen Arbeitsorganisation tatsächlich gearbeitet wird und welche physischen und sozialen Bedingungen am Arbeitsplatz herrschen, kann oft nur im direkten Kontakt mit dem eigentlichen Anwender erfasst werden (Beyer & Holtzblatt 1998, S. 42f., S. 113). Die Methoden des benutzerzentrierten Gestaltens können sicherstellen, dass die Sicht und die Interessen der Anwender ausreichend in den Business Use Cases repräsentiert werden.

Eine weitere wesentliche Rolle in der Geschäftsprozessmodellierung ist deshalb der **Nutzungskontextanalytiker**, in dessen Aufgabenbereich die Sammlung von Informationen über die Nutzer, weitere Interessensvertreter, Aufgaben der Nutzer sowie die physische, technische, organisatorische und soziale Umgebung aus Dokumenten und empirischer Datenerhebung gehört.

5.3 Gestaltung der Benutzungsoberfläche im Requirements-Workflow

Nach RUP wird die iterative Gestaltung der Benutzungsoberfläche von den ersten Konzepten bis hin zur Spezifikation und zum Styleguide dem Requirements-Workflow zugeordnet (Kruchten, Ahlquist, Bylund 2001). Der iterative benutzerzentrierte Gestaltungsprozess muss vor Beginn der Entwicklungs-Iterationen abgeschlossen sein. Erst wenn wesentliche Aufgabenabläufe gestaltet und evaluiert sowie die Usability-Ziele erreicht wurden, dann sollten die Iterationen der Entwicklung unter Einbindung von Benutzungsoberflächenaspekten starten. Somit erstreckt sich der iterative Gestaltungsprozess vor allem auf die Inception und die Elaboration Phase.

Der RUP sieht für den Requirements-Workflow folgende Rollen vor:

- Der **Systemanalytiker** erarbeitet zusammen mit den Interessensvertretern des Projekts, was das System leisten soll (und was es nicht leisten soll), und identifiziert die nichtfunktionalen Anforderungen. Darauf aufbauend entsteht die Vision für das zu entwickelnde System – dargestellt als eine Menge von Eigenschaften aus der Sicht der Interessensvertreter. Das Visionsdokument ist dann die Basis zur Entwicklung eines Use Case Modells.
- Der **Use Case Specifier** detailliert die Use Cases und bringt diese in Einklang mit den anderen Artefakten des Anforderungs-Workflows.
- Der **User Interface Designer** arbeitet parallel an der Definition des User Interface. Neben dem Design der Benutzungsoberfläche ist er für die Erstellung von Use Case Storyboards und des UI Prototypen zuständig (Kruchten, Ahlquist, Bylund, 2001). Diese Rolle wird im Hinblick auf die Integration von Usability-Engineering noch weiter differenziert werden.
- Der **Architekt** ist innerhalb des Anforderungsworkflow für die Identifizierung der Use Cases verantwortlich, deren Inhalt für die Architektur wichtig ist, und trägt zu deren Definition bei.

Um Entwicklungsprojekte mit einem Usability-Engineering integrierenden Prozess abzuwickeln, müssen in die Prozessplanung zusätzlich zu den im RUP definierten Rollen weitere hinzugefügt und die des User Interface Designers weiter unterteilt werden. In Anlehnung an Göransson, Lif, & Gulliksen (2003) und dem RUP User Experience Plug-In von IBM (2003) werden weitere Rollen vorgeschlagen:

- **Nutzungskontextanalytiker**
  Der Usability Engineer, der bereits über die Nutzungskontextanalyse am Requirements-Engineering beteiligt ist, detailliert – zusammen mit dem Use Case Specifier – die Use Cases.
- **User Interface Designer**
  Diese Rolle ist im RUP bereits vorhanden. Dort wird sie in erster Linie so verstanden, das User Interface Designer auf der Basis der Use Cases arbeitet. Aufgrund der Komplexität des User Interface Designs im Rahmen Benutzerzentrierter Gestaltung, ist es sinnvoll den User Interface Designer weiter zu präzisieren:
  - **Information Architect oder Interaction Designer**
    Die Aufgabe liegt darin, aus den erhobenen Informationen und Daten eine Informationsarchitektur bzw. ein konzeptuelles Modell der Benutzungsoberfläche zu entwerfen. Ferner werden Navigation, Dialoge und Interaktionen sowie Informationsdarstellungen unter Einbeziehung von Normen, Standards und Regeln und Erkenntnissen der HCI gestaltet.
  - **Graphikdesigner**
    Graphikdesigner entwickeln und definieren die Visuals, die Gesamtanmutung sowie die Umsetzung des Corporate Designs.
  - **Prototyper**
    Dieser Experte hat die Aufgabe, die Entwürfe der Benutzungsoberfläche erfahrbar und kommunizierbar zu machen. Der Prototyper wählt je nach Projektphase (z.B. Prototyp als

Basis für empirische formative Evaluation), Usability-Zielen, Kommunikationszielen (z.B. Demonstration des Produktes beim Produktmanagement oder Marketing) die passende Prototyping-Methode (z.B. Papierprototyp, Wizard of Oz Prototyp) aus und entwickelt die erforderlichen Prototypen.

- **Usability Evaluator** für formative Evaluation im Rahmen iterativer benutzerzentrierter Gestaltung
  Formative Evaluationen, welche die User Interface Designer bei ihrer Gestaltungstätigkeit unterstützen, werden vom Usability Evaluator geplant, durchgeführt, ausgewertet und kommuniziert.
- **Spezifikations- und Styleguide-Redakteur**
  Mit dieser Rolle werden die Aufgaben bezeichnet, die sicherstellen, dass das Benutzungsoberflächenkonzept vollständig, verständlich und präzise dargestellt wird (vgl. Görner, 2003, S. 160).

Zu den genannten Rollen können je nach Projekt weitere hinzugefügt werden. Bei Benutzungsoberflächen mit Sprachein- und -ausgabe kann es beispielsweise sinnvoll sein für diese spezielle Interaktionsform eine eigene Rolle zu definieren. Armitage (2003) entwickelte ein Disziplinenmodell für die Gestaltung von Benutzungsoberflächen interaktiver Produkte. Dort kann die Disziplinenverteilung bei der Oberflächengestaltung entnommen werden. Bei kleineren Projekten werden die verschiedenen Rollen von weniger Personen wahrgenommen, während es bei großen Projekten sogar Teams für einzelne Rollen geben kann. Wichtig jedoch ist, dass Personen, die diese Rollen ausfüllen sollen, ausreichend in Usability-Engineering und Human-Computer Interaction ausgebildet sind. Gulliksen, Göransson und Lif (2001, S. 294) schreiben dazu: „our experience, based on observations of several projects applying RUP in the user interface design process, is that successful adoption of user-centered design with RUP is the result of the participation of a person with high skills and experience in human-computer interaction".

5.4 Usability-Engineering in der Konstruktionsphase

Die Erfahrung aus zahlreichen Usability-Engineering-Projekten hat gezeigt, dass Spezifikationen der Benutzungsoberfläche meist nicht vollständig sind. Bei der Implementierung tauchen spezielle Fragen auf, die erst durch die Umsetzung in Software deutlich werden. Dazu gehört beispielsweise die Definition von seltenen Interaktionssituationen, wie z.B. welche Reaktion die Oberfläche zeigt, wenn ein Eingabefeld bereits die maximale Anzahl von Zeichen enthält, aber weitere Zeichen vom Nutzer eingegeben werden. Hier müssen Gestaltungsentscheidungen getroffen werden, die eindeutig in den Bereich des Usability-Engineerings fallen. Zur Unterstützung der Entwicklungsteams sollten dann speziell für diese Fragen ein Interaktionsdesigner hinzugezogen werden. Ist das Interaktionskonzept entsprechend verfeinert worden, dann muss die Spezifikation und/oder der Styleguide entsprechend der neuen Entscheidungen präzisiert werden. Hier sind als Rollen wiederum der Interaktionsdesigner sowie der Spezifikations- und Styleguide-Redakteur involviert. Personen, denen diese Rollen zugeordnet sind, können in die Entwicklungsteams integriert werden, was bei umfangreichen Projekten angemessen ist. Bei kleineren Projekten hat es sich bewährt, dass die Entwicklungsteams Gestaltungsentscheidungen, welche die Usability des Produkts betreffen, sammeln und in Gestaltungsworkshops gemeinsam mit den Interaktionsdesignern sowie den Redakteuren entscheiden.

5.5 Testen und Evaluieren

Zielsetzung des RUP Test-Workflows ist die Überprüfung des Interaktionsverhaltens der einzelnen Komponenten, ihrer ordnungsgemäßen Integration in das System sowie, dass Anforderungen korrekt implementiert wurden. Es liegt in der Verantwortung des Test-Workflows sicherzustellen, dass alle in den Tests entdeckten Defizite erfasst und adressiert sind, bevor ein System bereitgestellt wird. Der RUP Test-Workflow umfasst Aktivitäten zur Qualitätsüberprüfung, aber nicht zur Qualitätssicherung. Es ist die Aufgabe der Tester auf Mängel hinzuweisen; geeignete Maßnahmen zur Behebung der Mängel zu ergreifen ist dagegen Aufgabe des Projektteams. Das Methodenrepertoire des RUP ist ausgerichtet auf die Überprüfung der Qualitätskriterien: Zuverlässigkeit, Funktionalität und Leistungsverhalten. Getestet wird in insgesamt vier Stufen von der einzelnen Komponente (Unit Test) bis hin zum vollständigen System (Akzeptanztest). Während die ersten drei Teststufen vor allem darauf ausgerichtet sind die genannten Qualitätskriterien zu überprüfen, soll in der letzten Stufe die Akzeptanz des vollständigen Systems mit Benutzern bzw. Benutzervertretern überprüft werden, bevor darüber entschieden wird, ob das System einsatzbereit ist. Für die Testdurchführung stehen verschiedene Arten von Tests zur Verfügung, die auf einzelne Softwarequalitätsmerkmale spezialisiert sind. Die hauptverantwortlichen Rollen im Test-Workflow sind die des Testdesigners und des Testers, wobei der Testdesigner für die Erstellung eines Testplans und die Analyse der Testergebnisse verantwortlich ist, der Tester hingegen die einzelnen Tests durchführt und deren Ergebnisse zur Verfügung stellt.

Die Eigenschaften und Zielsetzung des RUP Test-Workflows machen deutlich, dass die darin verankerten Tests nicht dazu geeignet sind konstruktiv zum Gestaltungsprozess beizutragen, sondern Softwarequalität in unterschiedlichen Dimensionen zu überprüfen. Aus Sicht des Usability-Engineering sind hierzu summative Evaluationsverfahren geeignet, die am Ende des Entwicklungsprozesses zum Einsatz kommen und z.B. zur Überprüfung der Normenkonformität wie der ISO 9241 dienen (z.B. DaTech-Prüfhandbuch 2001). Hingegen dienen formative Evaluationsverfahren, wie z.B. Usability-Tests mit Benutzern, dem Auffinden von Usability-Problemen und der Optimierung von Gestaltungslösungen. Von daher sind diese Evaluationsverfahren eng verknüpft mit dem Entwurf von Benutzungsoberflächen und in einem RUP-basierten Entwicklungsprozess Bestandteil des Requirements-Workflows sowie des Analyse- und Design-Workflows. Die Akzeptanzüberprüfung im Sinne des Test-Workflows sollte in der Verantwortung einer auf Usability-Fragen spezialisierten Rolle liegen, der Rolle des Usability-Evaluators. Dieser entscheidet über die geeigneten Testverfahren zur Akzeptanzprüfung, organisiert deren Durchführung und die Auswertung der Ergebnisse. Im Rahmen des Test-Workflows weist er auf mögliche

Risiken hin (Qualitätsüberprüfung), mögliche Konsequenzen hieraus und die Entscheidung auf Nachbesserungen oder eine Verschiebung der Systemfreigabe (Qualitätssicherung) müssen vom Projektteam getroffen werden.

5.6 Supporting Workflow

Für die Begleitung eines Projekts durch alle Entwicklungsphasen gibt es im RUP weitere Workflows, die auf die Projekt- und Prozessunterstützung zielen. Hierzu zählen Konfigurationsmanagement, Projektmanagement und Entwicklungsumgebungsmanagement. Auch zu diesen Workflows kann Usability Engineering wertvolle Unterstützung leisten, angefangen von der Analyse und Gestaltung der Total User Experience bis hin zur Auswahl geeigneter, integrierter Entwicklungsumgebungen, die beispielsweise Möglichkeiten zur schnellen Erstellung von Benutzungsoberflächen-Prototypen und deren Wiederverwendung bieten.

Im Sinne eines benutzerzentrierten Vorgehensmodells und zur Etablierung der Nutzungsqualität als wichtiges Qualitätsmerkmal im Prozess und Unternehmen sollte, wie es auch im UCD-Prozess der IBM (IBM 2004) vorgeschlagen wird, ein User Experience Manager, oder Manager für Gebrauchstauglichkeit, alle Usability relevanten Aktivitäten von der Nutzungskontextanalyse bis hin zur Akzeptanzüberprüfung koordinieren, Usability-Ziele setzen und verfolgen, und die Verantwortung für das Management der Usability-Engineering-Maßnahmen übernehmen.

Bei einem derart umfangreichen Rahmenwerk wie dem Rational Unified Process ist es eher selten der Fall, dass alle im RUP vorhandenen Abläufe und Aktivitäten während der verschiedenen Projektphasen zum Tragen kommen. Mit jedem neuen Projekt sollte bei Beginn des Projekts auf Grundlage der Projektrahmenbedingungen festgelegt werden, welche Aktivitäten für das Projekt sinnvoll eingesetzt werden können und wie diese ggf. angepasst werden müssen. So sollte der RUP auch nicht als fertiges Produkt „out of the box" verstanden werden, sondern als Angebot, aus dem das jeweils Passende herausgesucht und adaptiert werden kann und sollte. Gulliksen, Göransson und Lif merken aus ihren Untersuchungen zum Einsatz von RUP in Entwicklungsprojekten dazu an „the person responsible for applying RUP to user interface design thought that the model contained answers to all problems and stopped acting as a thinking person" (2001, S. 294).

6. Einbindung in die Projektpraxis

Wenn Usability ein Gestaltungsziel bei der Entwicklung eines Systems ist, dann muss Usability-Engineering konsequent in den jeweiligen Software-Engineering-Prozess integriert werden. Inhalte und Umfang von Workflows verändern sich daraufhin entsprechend. Die Integration von Usability-Engineering-Maßnahmen ergänzt systematisch und methodisch ohnehin notwendige Aktivitäten des Software-Engineerings.Nutzungskontextanalyse unterstützt die Modellierung von Geschäftsprozessen und das Requirements-Engineering. Eine Benutzungsoberfläche muss ohnehin entworfen und gestaltet werden. Auf Basis einer sorgfältigen Nutzungskontextanalyse und mit dem notwendigen Sachverstand von Usability-Engineering-Experten lässt sich die Nutzungsqualität eines Systems auf jeden Fall deutlich verbessern.

Die Herausforderung für das Projektmangement liegt darin, Usability-Engineering nicht als separaten Prozess oder punktuelle Maßnahme einzuplanen. Vorhandene Softwareentwicklungsprozesse sollten erweitert werden, dass Benutzerzentrierte Gestaltung als treibende Kraft ein Teil des gesamten Prozesses wird. Anregungen dazu haben wir hier am Beispiel des RUP gezeigt.

Um Usability-Engineering- und Software-Engineering-Prozesse zu verzahnen, müssen in Zukunft aus unserer Sicht folgende Punkte stärker betrachtet werden:

- Stärkere Integration der Ergebnisse von Usability-Engineering-Maßnahmen in die etablierten Modellierungsstrategien. Ansätze hierzu finden sich bei beispielsweise bei Gulliksen, Göransson und Lif (2001), Kruchten, Ahlquist und Bylund (2001) Phillips und Kemp (2002), sowie in diesem Artikel.
- Methoden wie Contextual Inquiry (Beyer und Holtzblatt 1998) oder User Testing, die die Ausgangsinformationen für Modelle liefern, müssen stärker integriert werden.
- Definition von operationalisierbaren Usability-Zielen als Bestandteil der nicht-funktionalen Anforderungen, die insbesondere auch als Grundlage für das Testing dienen.
- Integration der für Usability-Engineering erforderlichen Rollen in die Prozessstruktur.
- Insgesamt eine stärkere Verpflichtung zur Usability bei der Entwicklung von Systemen mit einem großen Anteil an Nutzerinteraktion und eine konsequente Ausrichtung auf Benutzerzentrierte Gestaltung.

Wir sind überzeugt davon, dass eine konsequente Prozessintegration von Software- und Usability-Engineering zu einer effizienteren Vorgehensweise bei der Produktentwicklung führt und Reibungsverluste in der Umsetzung von Resultaten aus Usability-Maßnahmen in das aktuelle Produktdesign minimieren hilft. Entsprechend der großen Breite von Projektdimensionen und Anforderungen kann es auch nicht den allein gültigen Prozess geben, sondern nur einen Prozessrahmen und ein Angebot unterschiedlicher Rollen, Verantwortlichkeiten und Methoden. Aufgabe der Projektinitialisierung und des Projektmanagements ist es, hieraus das Geeignete und Notwendige auszuwählen und auf die jeweiligen Projekterfordernisse und Situationen so anzupassen, dass Benutzerzentrierte Gestaltung zu Produkten mit einer hohen Usability-Qualität führt.

Literatur

Armitage, J.: From User Interface to Über-Interface: A design discipline Model for digital products. *Interactions* **X(3)** (2003) 21–29.

Beyer, H.; Holtzblatt, K.: *Contextual Design- Defining Customer-Centered Systems*. San Francisco: Morgan Kaufmann Publishers, 1998.

Bias, R.; Mayhew, D.: *Cost Justifying Usability*, New York: Academic Press, 1994.

Cockburn, A.: *Writing Effective Use Cases*, Boston: Addison Wesley, 2001.

DATech: *DATech-Prüfhandbuch Gebrauchstauglichkeit Version 3.2*, DATech Deutsche Akkreditierungsstelle Technik e.V., 2001.

DIN EN ISO 9241-11: *Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten – Teil 11: Anforderungen an die Gebrauchstauglichkeit; Leitsätze* (ISO 9241-11:1998). Berlin: Beuth, 1998.

DIN EN ISO 13407: *Benutzerorientierte Gestaltung interaktiver Systeme*, Berlin: Beuth Verlag, 2000.

Geissbühler, M.; Hassenzahl, M.; Beu, A.: Das Usability Service Model der Swiss Re. In: *User Interface Tuning – Benutzungsschnittstellen menschlich gestalten* (Hrsg. Machate, J.; Burmester, M.). Frankfurt: Software und Support, 2003.

Göransson, B.; Lif, M.; Gulliksen, J.: Usability Design – Extending Rational Unified Process

with a New Discipline. In: *Interactive Systems: Design, Specification and Verification.* (Hrsg. Nunes, J.; Cunha, N. J.) Berlin: Springer-Verlag, 2003.

Görner, C.: Styleguides – Vom Ladenhüter zum Steuerungsinstrument. In: *User Interface Tuning – Benutzungsschnittstellen menschlich gestalten.* (Hrsg. Machate, J.; Burmester, M.) Frankfurt: Software und Support, 2003.

Gould, J. D.; Lewis, C. H.: Designing for usability: key principles and what designers think. *Communications of the ACM* **28[3]** (1985) 300–311.

Gulliksen, J.; Göransson, B.; Lif, M.: A User Centred Approach to Object-Oriented User Interface Design. In: *Object Modeling and User Interface Design*. (Hrsg. van Harmelen, M.) Boston: Addison-Wesley, 2001.

Hassenzahl, M.; Burmester, M.: Zur Diagnose von Nutzungsproblemen: Praktikable Ansätze aus der qualitativen Forschungspraxis. Konferenzband des ZMMS Konferenz 5.10 – 8.10.1999 in Berlin, 1999.

Hix, D.; Hartson, H. R.: *Developing user interfaces: ensuring usability through product and process.* New York: John Wiley, 1993.

IBM: User Engineering. http://www-306.ibm.com/ibm/easy/eou_ext.nsf/publish/1996 (Letzter Zugriff: 04.08.2005).

IBM: RUP Plug-Ins for User Experience. http://www-128.ibm.com/developerworks/rational/library/4310.html#v3 (Letzter Zugriff: 13.10.2005).

John, B. E.; Bass, L.; Adams, R. J.: Communication across the HCI/SE divide: ISO 13407 and the Rational Unified Process®. In: *Proceedings of HCI International*, June 2003.

Karat, C. M.: Cost-justifying Usability-Engineering in the software life cycle. In: *Handbook of Human-Computer Interaction.* (Hrsg. Helander; M.; Landauer; T.; Prabhu, P.) Amsterdam: Elsevier Science, 1997.

Kruchten, P.: *The Rational Unified Process – An Introduction*, Second Edition, :Ort: Addison Wesley Longman, 2000.

Kruchten, P.; Ahlquist, S.; Bylund, S.: User Interface Design in the Rational Unified Process. In: *Object Modelling and User Interface Design*. (Hrsg. van Harmelen, M.) Boston: Addison-Wesley, 2001.

Mayhew, D. L.: *The Usability-Engineering lifecycle. A practitioner's handbook for user interface design*. San Francisco, CA: Morgan Kaufmann, 1999.

Nielsen, J.: *Usability-Engineering* . Boston, San Diego: Academic Press, 1993.

Phillips, C.; Kemp, E.: In Support of User Interface Design in Rational Unified Process. In: *Conferences in Research and Practice in Information Technology*. (Hrsg. Grundy, G.; Calder, P.) Vol. 7. Proceedings of the Third Australian User Interface Conference (AUIC 2002, pp. 21-27), Melbourne, Australia, 2002.

Raskin, J.: *Das Intelligente Interface*. München: Addison-Wesley, 2001.

Redish, J.; Wixon, D.: Task Analysis. In: *The Human-Computer Interaction Handbook - Fundamentals, Evolving Technologies and Emerging Applications.* (Hrsg. Jacko, J. A.; Sears, A.) Mahwah, N.J.: Lawrence Erlbaum Associates, Publ., 2003.

Rosson, M. B.; Carroll, J. M.: *Usability-Engineering – Scenario-based development of human-computer interaction*. San Francisco: Morgan Kaufmann, 2002.

RUP IBM: *RUP Ressources*. http://www-136.ibm.com/developerworks/rational/products/rup (Letzter Zugriff: 04.08.2005).


1


2


3

1 Prof. Dr. Michael Burmester, Dipl.-Psych., bis Ende 1996 als wissenschaftlicher Mitarbeiter am Fraunhofer-Institut für Arbeitswirtschaft und Organisation (IAO) in Stuttgart, danach im Fachzentrum User Interface Design der Siemens AG. Von 2000 bis 2002 leitete er den Bereich Usability-Engineering und die Geschäftsstelle München der User Interface Design GmbH und übernahm anschließend die Professur für Ergonomie und Usability an der Hochschule der Medien in Stuttgart. Er verfügt über langjährige Erfahrung als Leiter von nationalen und internationalen Forschungs- und Beratungsprojekten. Hauptarbeitsgebiete: Methoden benutzerzentrierter Gestaltung, Attraktivität interaktiver Produkte.
E-Mail: michael.burmester@uidesign.de; Burmester@hdm-stuttgart.de
www.user-interface-tuning.de;

2 Dr. Joachim Machate, Dipl.-Inform., nach Tätigkeiten am Fraunhofer IAO und bei der IBM Deutschland Informationssysteme GmbH seit 2000 Leiter Software-Entwicklung und Manager IT & Enterprise Solutions bei der User Interface Design GmbH. Er beschäftigt sich mit der Integration von Software- und Usability-Engineering und der Optimierung bei der Dokumentation des Entwurfs- und Spezifikationsprozesses von Benutzungsschnittstellen. Als Lehrbeauftragter für Software-Ergonomie & Design unterrichtet er an der Fachhochschule Heilbronn sowie als Referent für Benutzerzentriertes Gestalten interaktiver Produkte an der Technischen Akademie Esslingen.
E-Mail: joachim.machate@uidesign.de
www.uidesign.de; www.user-interface-tuning.de

3 Nina Sandweg, Dipl.-Inform., hat ihr Diplom in Informatik (Nebenfach Theoretische Medizin) an der TU München gemacht. Von 1999 bis 2004 arbeitete sie als Principal Consultant im Fachzentrum User Interface Design der Corporate Technology der Siemens AG mit einem Arbeitsschwerpunkt auf der Gestaltung von User Interfaces im Bereich der Medizintechnik und der Siemens-weiten Integration von Usability-Engineering-Methoden in Marketing- und Produktentwicklungsprozesse. Seit 2005 verantwortet sie als Produktmanagerin bei der Siemens Audiologische Technik GmbH das Thema User Interface Design.
E-Mail: nina.sandweg@siemens.com