Die benutzerzentrierte Entwicklung mobiler Unternehmenssoftware

Prof. Dr.-Ing. Thomas Ritz

Fachhochschule Aachen
52066 Aachen
ritz@fh-aachen.de

Abstract: Der vorliegende Artikel beschreibt eine aus agilen Entwicklungsmethoden und Usability Engineering zusammengesetzte Methode zur Entwicklung mobiler Unternehmenssoftware. Er führt dabei allgemein in das Problemfeld ein und schließt mit einer Bewertung der hier vorgestellten Arbeiten.

1 Einleitung

Im industriellen Berufsalltag finden sich zunehmend Menschen, die nicht mehr an einem festen Standort arbeiten. Die Einbindung dieser „mobile Workforces“ in betriebliche Anwendungssysteme ist eine der aktuellen Herausforderungen für Softwareentwickler. Obwohl die technischen Randbedingungen wie Netzwerke, Endgeräte, aber auch die Entwicklungsumgebung zunehmend stabiler werden, zeigen sich in vielen Projekten Schwierigkeiten bei der Entwicklung akzeptierter und nutzenbringender mobiler Anwendungen. Dieser Beitrag beschäftigt sich mit dem methodischen Vorgehen bei der Entwicklung mobiler Unternehmenessoftwaresysteme. Dazu wird zunächst der Begriff der mobilen Unternehmenssoftware definiert. Darauf aufbauend werden Probleme und Besonderheiten bei der systematischen Entwicklung solcher Systeme aufgezeigt. Schließlich wird ein Ansatz präsentiert, der diese Besonderheiten durch die Integration von Extreme Programming und Usability Engineering aufgreift und damit das benutzerzentrierte Entwickeln mobiler Unternehmenssoftware ermöglicht. Der Artikel schließt mit einer kritischen Betrachtung der vorgeschlagenen Methode.

2 Der Begriff „Mobile Unternehmenssoftware“

Der Begriff Unternehmenssoftware hat sich als unscharfe Zusammenfassung von betrieblichen Anwendungssystemen etabliert. Zusammengefasst werden damit etwa Anwendungen zur Verwaltung betrieblicher Ressourcen (ERP oder auch HR), Informationssysteme für Marketing und Vertrieb (CRM), aber auch Informationssysteme, die direkt der Produktion (etwa Produktionsplanung und Steuerung (PPS)) oder Konstruktion (etwa CAD) dienen. Viele dieser in Prozesse integrierten Systeme erheben, veredeln oder nutzen zentral gespeicherte Informationen, die unternehmensweit zur Verfügung stehen oder zur Verfügung gestellt werden. Die

Einbindungen von Mitarbeitern, die nicht an einem festen Standort arbeiten, ist das Ziel von mobiler Unternehmenssoftware. Die nachfolgende Definition von mobiler Unternehmenssoftware (vgl. [Rit2003]) erweitert diese examplarische Vorstellung um wichtige Anforderungen:

„Mobile Unternehmenssoftware ist die Anwendung von Unternehmenssoftware im mobilen Einsatz

1. auf adäquaten mobilen Endgeräten

2. mit angepasster Funktionalität

3. basierend auf Daten von adäquater Aktualität. „

Während sich die Anpassung der Funktionalität und die Auswahl einer adäquaten Hardware aus den veränderten Interaktionsparadigmen ((vgl. etwa [Sal2005]) bzw. der Entwicklung von miniaturisierter Hardware (vgl. [KR2002])) unmittelbar ergibt, hat sich gerade die dritte geforderte Eigenschaft als in der Praxis häufig vernachlässigte Anforderung gezeigt. Daten können lokal auf dem mobilen Computer gehalten werden oder auch über diverse Netzwerktechnologien abgerufen werden (vgl. etwa [Han2003]). Um hier eine geeignete Architektur festzulegen, hilft es die Unterschiede zu „stationären Anwendungen“ zu reflektieren. Bei stationären Computern ist die Verfügbarkeit einer Netzwerkinfrastruktur konstant. Im mobilen Bereich ist die Verfügbarkeit von Netzwerkinfrastruktur jedoch in einem hohen Maße abhängig von der Örtlichkeit, an der gearbeitet wird und diese ist im Allgemeinen nicht konstant. Darüber hinaus sind die unterschiedlichen Netzwerke, die mobil verfügbar sein können, häufig mit Kosten der Nutzung belegt. Diese Überlegungen machen deutlich, dass für den Betrieb mobiler Unternehmenssoftware bewertet werden muss, wie aktuell Informationen sein müssen und wie „veraltet“ sie sein dürfen. Zentral ist hierbei oft die Frage: Welche Daten müssen online abgerufen werden, weil die Informationen ansonsten veraltet sind, und welche können dann abgeglichen werden, wenn das mobile Computersystem sicher über Netzwerkkonnektivität verfügt.

3. Die Entwicklung mobiler Unternehmenssoftware

Neben technischen Aspekten treten Herausforderungen im methodischen Vorgehen bei der Entwicklung mobiler Unternehmenssoftware („Software Engineering“) auf. Die Besonderheiten können anhand des im Software Engineering eingehend betrachteten Wasserfallmodells deutlich gemacht werden (vgl. z.B. [Bal2001]). Im Folgenden werden die Phasen Anforderungsanalyse, Spezifikation, Entwicklung und schließlich Test und Betrieb näher betrachtet.

Im Rahmen der Anforderungsanalyse bildet die Prozessanalyse samt einer Prozessmodellierung häufig die Grundlage für das weitere Vorgehen. Betrachtet man die gängigen Methoden zur Geschäftsprozessmodellierung wird deutlich, dass Abläufe, auslösende Ereignisse, Prozessträger sowie die beteiligten DV-Systeme als Modellierungsobjekte zur Verfügung stehen. Für die Entwicklung von mobiler Unternehmenssoftware wichtige Inputfaktoren, wie den Ort der Verrichtung für einzelne Prozessschritte, sowie die dort verfügbare Infrastruktur, werden dabei nicht berücksichtigt. Erste Ansätze liefern hier Methoden wie das Mobile Process Landscaping (vgl. [KG2004]) oder auch das Mobile Process Blueprinting (vgl. [SR 2003]). Hier werden für die einzelnen Prozessschritte Lokationen und zumindest bezüglich der Netzwerke Infrastrukturmerkmale modelliert. Die Modellierung von Arbeitsbedingungen (etwa Lichtverhältnisse oder Lautstärke) werden bisher kaum methodisch fundiert betrieben.

Durch eine weitestgehende Harmonisierung der Infrastrukturen zum Betrieb von Unternehmenssoftwaresystemen ist im Rahmen der Spezifikation die technische Spezifikation in den Hintergrund getreten. Funktionale Aspekte stehen im Vordergrund. Bei der Entwicklung mobiler Unternehmenssoftware ist dieser Umstand nicht haltbar, da wie bereits gesehen die technischen Möglichkeiten sehr heterogen sind. Auch ist die Umsetzung der bei mobiler Unternehmenssoftware neu auftretenden Anforderungen (etwa Arbeit unter wechselnden Lichtverhältnissen) durch eine entsprechende Spezifikation sicherzustellen. Schließlich ist in der Spezifikation klar herauszustellen, welche technischen Komponenten Dritter verwendet werden (etwa öffentliche Mobilfunknetze). Daraus entstehende Implikationen hinsichtlich Sicherheit sollen an dieser Stelle nur erwähnt werden.

Bei der eigentlichen Entwicklung ist zu beachten, dass diese häufig nicht auf der spezifizierten Zielplattform durchgeführt wird. Zum schnellen Testen der Applikationen bieten die meisten Entwicklungsumgebungen daher entsprechende Emulatoren (vgl. z.B. Abbildung 1). Auf diesen Emulatoren kann lediglich getestet werden, ob spezifizierte Funktionen umgesetzt wurden. Die tatsächliche Benutzbarkeit bleibt jedoch wegen der am PC großzügig vorhandenen Interaktionsmöglichkeiten verborgen und bedingt einen regelmäßigen Test auf dem mobilen Endgerät.

Einige Softwareentwicklungsumgebungen versuchen der Vielzahl an verfügbaren Endgeräten Rechnung zu tragen, indem die Benutzerschnittstellen an die technischen Möglichkeiten des Ausgabegerätes automatisch zur Laufzeit angepasst werden (vgl. etwa Microsoft Mobile Controls[1]). Damit werden aktuelle Methoden zur Entwicklung von GUIS mittels WYSIWYG (What You See Is What You Get) obsolet. Neueste Ansätze gehen dabei davon aus, dass in Zukunft Benutzerschnittstellen für mobile Anwendungen aus entsprechenden generischen Dialogbeschreibungssprachen erzeugt werden (vgl. etwa [Ali2001] oder auch [SR2001]).

[1] http://www.asp.net/mobile/intro.aspx?tabindex=3&tabID=44

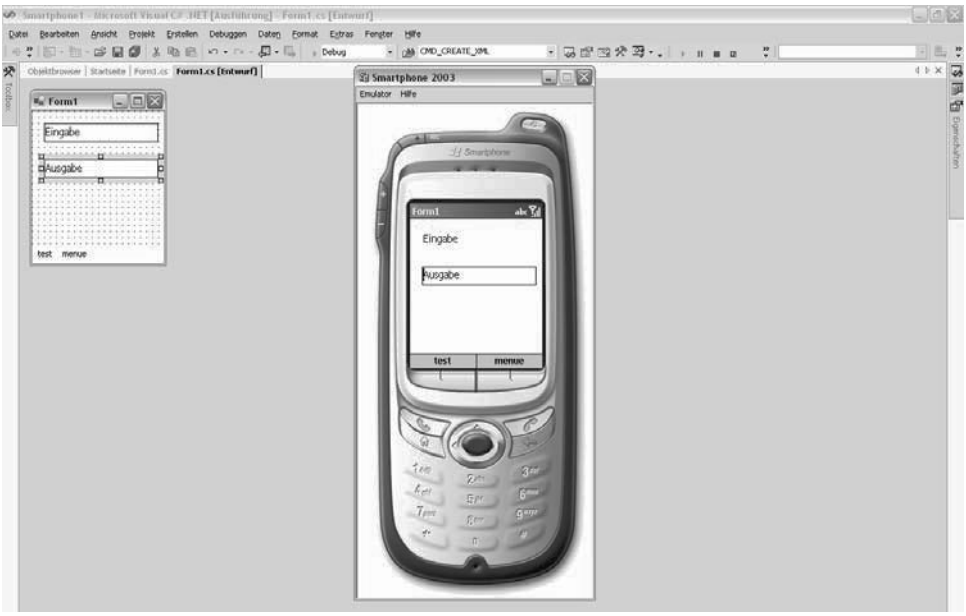

Abbildung 1: Geräteemulator in Microsoft Visual Studio

Bei der Entwicklung stationärer Software kann die Testphase ortsunabhängig durchgeführt werden. Dies gilt für mobile Unternehmenssoftware nicht. Hier entsteht die Schwierigkeit, die Software entweder an den späteren Einsatzorten zu testen, oder Infrastruktur und Umgebungsbedingungen im Labor bestmöglich nachzustellen. Das Nachstellen von Infrastrukturkomponenten von Drittanbietern (etwa UMTS-Empfang) zeigt sich dabei im Labor als besonders schwierig. Die Einführung von mobiler Unternehmenssoftware in Unternehmen zeigt zudem, dass für den Umgang mit der Hardware „mobiles Endgerät“ ein erheblicher Zeitanteil der Einführungsphase verplant werden muss.

4. Integration von Extreme Programming und Usability Engineering zur benutzerzentrierten Entwicklung von mobiler Unternehmenssoftware

Die vorangegangene Betrachtung hat in die Problematik bei der Entwicklung mobiler Unternehmenssoftware eingeführt und teilweise Lösungsvorschläge für auftretende Probleme geliefert. Für die folgenden Betrachtungen einer generalisierten Vorgehensweise soll vereinfachend davon ausgegangen werden, dass zur Entwicklung einer mobilen Unternehmenssoftware verschiedene Akteure agieren:

Entwickler (E)	Versierter Techniker/Programmierer, der die mobile Unternehmenssoftwaresysteme entwickelt
Außendienstmitarbeiter (AD)	Ein zukünftiger Nutzer der mobilen Unternehmenssoftware
Manager (M)	Entscheidungsträger des Unternehmens, die die mobile Unternehmenssoftware einsetzen

Um die nachfolgend aufgezählten Probleme einordnen zu können, ist wichtig vorauszuschicken, dass mobile Unternehmenssoftware ein innovativer, sich aber in ständiger, auch technischer, (Weiter-)Entwicklung befindlicher Markt ist. Als Indikator mag hier das Erscheinen von vier Betriebssystemgenerationen für Microsoft-PDA-Systeme im Vergleich zu einer Windows XP Generation über fünf Jahre dienen. Daraus resultieren folgende Problembereiche:

1. AD und M haben oftmals nur eine vage Vorstellung über die technischen wie organisatorischen Möglichkeiten durch den Einsatz mobiler Unternehmenssoftware und den notwendigen monetären, organisatorischen und anderweitigen Ressourcen.

2. E, selbst wenn er in einer Anwendungsbranche Expertise hat, tut sich schwer, der Anforderung aus der Definition von mobiler Unternehmenssoftware nach angepasster Funktionalität nachzukommen, da ihm die genauen Abläufe im Außendienst auch nach einer eingehenden Prozessanalyse nur skizzenhaft klar sein können, und darüber hinaus die genauen Arbeitsbedingungen (Umgebung) unklar sind. Dieses Wissen hat lediglich der AD.

3. AD und M haben oftmals konkurrierende Interessen bei der Einführung mobiler Unternehmenssoftware. Stehen beim AD Interessen im Vordergrund, die seine Arbeit einfacher bzw. schneller gestalten, ist oftmals das Interesse des M in einer besseren Überwachung der Außendienstmitarbeiter zu finden (dieser Aspekt wird hier nicht weiter betrachtet, weil Lösungen hier eher in der Anwendung von Change-Management-Ansätzen (vgl. etwa [Kot1997]) als in technischen Ansätzen zu suchen sind.)

Betrachtet man diese Unsicherheiten und vergleicht die dedizierten Anforderungen (insbesondere angepasste Funktionalität und adäquate Aktualität der Daten), die sich aus der Definition von mobiler Unternehmenssoftware ergeben, so ergeben sich folgende Anforderungen an ein Vorgehen bei der Entwicklung von mobiler Unternehmenssoftware:

1. Im Rahmen des Vorgehens müssen die Funktionalitäten (insbesondere die Interaktionen) in *enger Zusammenarbeit* mit dem AD entwickelt werden, da nur er über ein entsprechendes „Domainenwissen“ verfügt.

2. Möglichkeiten, die mobile Anwendungen bieten, müssen AD und M *transparent* gemacht werden, um so wiederum Input für eine optimale Gestaltung der Anwendung zu „provozieren“.

3. Die optimale Anpassung der Anwendung an die Bedürfnisse der AD sollte *messbar* gemacht werden.

Als Bewertungskriterium für den letzten Punkt kann dabei die Usability (Deutsch: Benutzbarkeit/Gebrauchstauglichkeit) herangezogen werden. Unter Usability versteht man den Grad der Effektivität, Effizienz und Zufriedenheit, mit der Benutzer vorher definierte Aufgaben mit einer Applikation in einem festgelegten Umfeld (etwa Büroumgebung mit definierter Ausstattung) ausführen können (vgl. [DIN2002]). In diesem Kontext haben sich Vorgehensmodelle etabliert, die systematisch Usability als Leistungsmaß über den gesamten Produktlebenszyklus betrachten. Diese Vorgehensmodelle werden unter dem Schlagwort „Usability Engineering“ zusammengefasst (vgl. etwa [May1999]).

Norman (vgl. [Nor1988]) schlägt schon 1988 für die Gestaltung von allgemeinen Gebrauchsgütern die Anwendung von ”benutzerzentrierten Ansätzen“ vor. Besonderes Augenmerk liegt dabei auf der Beachtung der kognitiven Prozesse der Nutzer. Nur mit einem tiefen Verständnis der kognitiven Prozesse lassen sich die mentalen Denkprozesse in Interaktionen am Rechner übertragen und garantiert so die Erwartungskonformität der Benutzerschnittstelle (vgl. [Nor1986]). Ein solcher benutzerzentrierter Ansatz, um die Gebrauchtauglichkeit von interaktiven Systemen sicherzustellen, wurde schließlich in DIN EN 13407 festgeschrieben. Methodische Ansätze zur benutzerzentrierten Gestaltung in den Phasen Analyse, Gestaltung, Prototyping und Evaluation sowie deren Bewertung finden sich etwa bei Burmester und Görner (vgl. [BG2003]). Diese beziehen sich jedoch lediglich auf die Gestaltung der Interaktion.

Um in Softwareprojekten den Kunden allgemein, also weit über die Interaktion hinaus, als wichtigen Inputfaktor einzubeziehen, haben sich agile Softwareentwicklungs-methoden (vgl. z.B. [Coc2001]) bewährt. Dabei werden die oben beschriebenen Phasen eines Softwareentwicklungsprojektes nicht länger sequentiell durchlaufen sondern iterativ. In diesen Iterationen wird der Kunde als wichtigster Inputfaktor sowohl zur Definition der Anforderungen als auch zur Abnahme von Teilprojekten herangezogen. Wichtige Vertreter dieser Entwicklungsmethoden sind Extreme Programming (vgl. z.B. [BA 2004]), Chrystal Methodologies (vgl. z.B. [Coc2005]) oder auch Scrum (vgl. z.B. [Sch2004].

Tabelle 1stellt einige wichtige Eigenschaften von Usabiltiy Engineering und Extreme Programming gegenüber.

	Usability (Engineering)	**Extreme Programming**
Evaluation	Quantitative und Qualitative Messung der	Testen mit Unit Test auf funktionale Anforderungen und

	Gebrauchstauglichkeit	anschließende Integration
Vorgehen	meist linearer Prozess in Schritten	Iterativer Prozess
Ansatz	User Centered	User Centered
Fokus	Interaktion im Vordergrund	Umsetzung allg. Benutzeranforderungen im Vordergrund (z.B. auch Anforderungen bzgl. Integration mit Fremdsystemen)
Pragmatische Vorgabe	Vorlage für Gestaltung	Vorlage für softwaretechnische Umsetzung (etwa mittels Refactoring, Pairprogramming)
Verwendung von Prototypen	Prototypen zur Visualisierung von Interaktionskonzepten, die nicht zwingend zum Produkt weiterverarbeitet werden	Prototypen, die eine Evolution zum Produkt durchlaufen („Refactoring“)
Projektmanagement	Keine/wenige Ansätze zur Verwaltung von Ressourcen	Ansatz zum Verwalten von Ressourcen durch Tasks und Storycards sowie Ideal Hours, Real Hours und Project Velocity

Tabelle 1: Usability Engineering vs. Extreme Programming (XP)

Im Folgenden wird eine Integration von Extreme Programming und Usability Engineering im Hinblick auf die Entwicklung von mobiler Unternehmenssoftware unternommen. Da beide Methoden einen benutzerzentrierten Ansatz vorsehen, wird die integrierte Methode als „benutzerzentrierte Entwicklung von mobiler Unternehmenssoftware“ bezeichnet. Diese Integration kann mit 12 Basiselementen beschrieben werden (vgl. Abbildung 2).

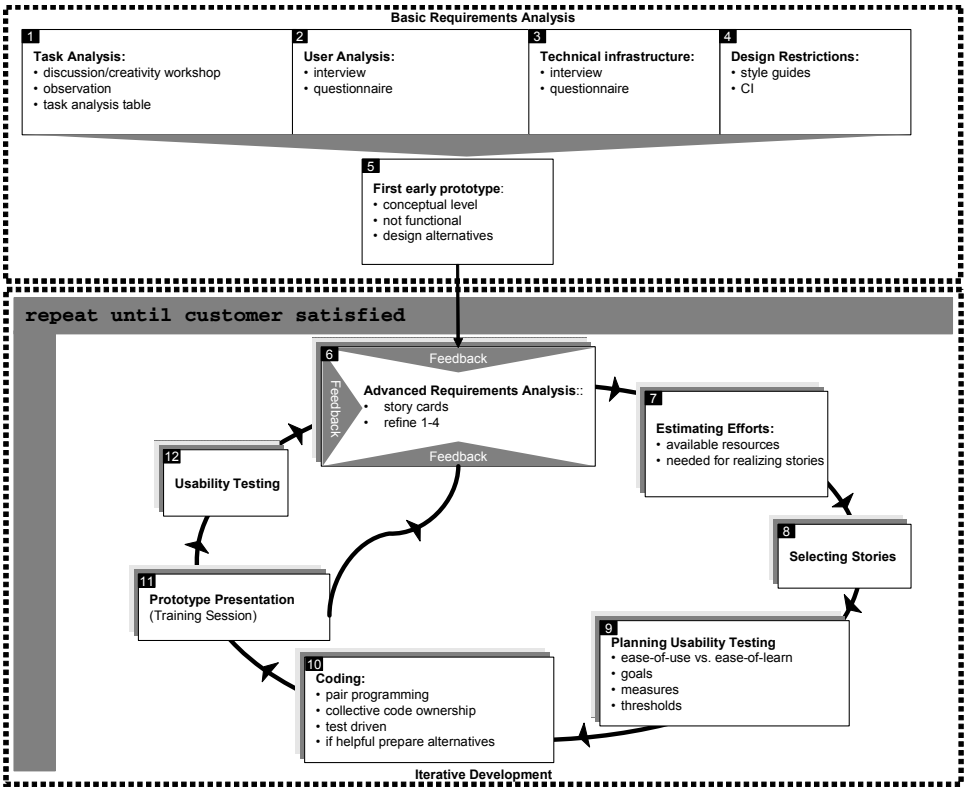

Abbildung 2: Integriertes Vorgehen

Zunächst lässt sich dieses Vorgehen in zwei Phasen unterteilen. In der ersten Phase wird eine generische Anforderungsanalyse unternommen, deren Schritte sich am „Usability Engineering Lifecycle“ von Mayhew orientieren (vgl. [May1999]). Diese resultiert in einem ersten frühen Prototypen, der stellvertretend für eine schriftliche Spezifikation den Start für die in der zweiten Phase folgenden Iterationen bildet.

Um die grundlegenden Arbeitsabläufe zu verstehen, wird zunächst eine oberflächliche Prozessanalyse betrieben (Schritt 1). Dabei werden bereits offensichtliche Probleme dokumentiert und Prozessbeteiligte ermittelt. Hierzu kommen hinlänglich bekannte Methoden der Prozessmodellierung, aber auch etwa die von Mayhew vorgeschlagene Task Analysis Table (vgl. [May1999]) in Frage. Bei der letzteren Methode werden tabellarisch folgende Informationen z.B. im Rahmen eines Kundenworkshops gesammelt:

- Akteure
- Auslösende Ereignisse
- Use Case

- Bekannte Probleme

Als Erweiterung werden im Hinblick auf die Mobilität die möglichen Verrichtungsorte bzw. Ortstypen der einzelnen Anwendungsfälle aufgenommen. Dazu kann entweder obiges tabellarisches Verfahren erweitert oder aber gängige Prozessnotationen um die von Kuhn et al. (vgl. [KG2004]) vorgeschlagenen Mobile Process Landscapes angereichert werden.

Für die Personengruppen, die von einer späteren IT-Lösung betroffen sind, werden detaillierte Benutzerprofile erhoben (Schritt 2). Auch hier kommen etablierte Methoden des Usability Engineerings zum Einsatz (spezielle Ansätze für mobile Anwendungen finden sich etwa in [GM2006] oder auch [Lov2005]). Wenn „wirkliche" Nutzer an dieser Stelle nicht greifbar sind, kann auch die Definition von Nutzergruppe (vgl. [Wei2002]) oder auch die Definition von „Personas" hilfreich sein (vgl. [GM2006]). Schließlich wird die technische Infrastruktur beleuchtet (Schritt 3). Besonders wichtig für mobile Anwendungen sind hierbei bereits etablierte Back-End-Systeme, in die integriert werden sollte. Hier gewonnene Informationen geben erste Hinweise auf eine sinnvolle Integrationsarchitektur. Schließlich werden bereits existierende Gestaltungsrichtlinien gesammelt und auf ihre Anwendbarkeit überprüft (Schritt 4). Neben allgemeinen Gestaltungsrichtlinien für mobile Anwendungen (vgl. etwa [Stu2005]), können auch produktspezifische Richtlinien (z.B. für SAP [SAP2003]) herangezogen werden. Basierend auf diesem rudimentären Wissen wird eine erste prototypische Lösung entwickelt. Diese muss nicht umfassend digital und interaktiv sein, sondern kann in Form von Zeichnungen etc. visualisiert werden (zur Anwendung von Prototypen bei mobilen Anwendungen vgl. [GM2006], [Lov2005] oder auch [Wei2002]). Die Praxis zeigt jedoch, dass eine digitale, aber nicht notwendigerweise interaktive, Form auf einem in Frage kommenden Endgerät dem Kunden ermöglicht, zu einem sehr frühen Zeitpunkt ein Gefühl für die Möglichkeiten und Probleme der mobilen Anwendung im Nutzungskontext zu entwickeln. Dieser sehr frühe Prototyp „regt die Phantasie" der Anwender an.

Die folgenden Iterationen werden durchgeführt, bis der Kunde mit dem Projektergebnis zufrieden ist, bzw. das Projektbudget erschöpft ist. Das Kundenfeedback auf den ersten Prototypen liefert den Input für die erste Iteration. Der Kunde, der nun eine bedeutend bessere Vorstellung über die Möglichkeiten von mobiler Unternehmenssoftware hat, als im Rahmen der Anforderungsanalyse, definiert nun Anwendungsfälle (XP Artefakt: „Story" vgl. [Wel2001]) (Schritt 6). Hierbei wird streng dem XP Paradigma gefolgt, dass der Kunde die Anwendungsfälle definiert. Im Rahmen von Workshops werden dazu die durch die generische Anforderungsanalyse ermittelten Use Cases vom Kunden erneut in seinen Worten niedergeschrieben. Diese Anwendungsfälle werden bereits durch die für mobile Unternehmenssoftware wichtigen Informationen wie Ausführungsort und Mengen-/Zeitgerüste für benötigte bzw. produzierte Informationen angereichert. Mengen-/Zeitgerüste werden dabei für mobil erstellte bzw. mobil konsumierte Informationen, wenn nötig auf Attributebene, erhoben. Die in Tabelle 2 aufgeführten Items haben sich dabei in diversen Projekten als zielführend herauskristallisiert.

Informationstyp	Beschreibung der Information
Format(e)	Welche Datenformate werden ausgetauscht? (insbesondere wichtig bei Formaten, die Breitbandtechnologien erfordern)
Änderung	Zyklische/Azyklische Änderung und deren Frequenz (Hinweise auf Replikations- bzw. Synchronisationsweg)
Häufigkeit des Abrufs	Wie oft wird auf das Attribut zugegriffen
Informationsmenge	Bestimmt durch Messung oder Schätzung
Wer pflegt die Informationen	In welcher Organisationseinheit bzw. welchem Informationssystem wird die Information gepflegt

Tabelle 2: Mengen/Zeit Gerüste

Die Anwendungsfälle werden durch die Entwickler insbesondere hinsichtlich Ihrer Entwicklungsaufwände bewertet und in Task Cards (vgl. [Wel2001]) überführt (Schritt 7) Der Kunde selektiert schließlich die „Stories“, die mit Hilfe der über die anstehende Iterationsdauer zur Verfügung stehenden Ressourcen umgesetzt werden können (Schritt 8). Basierend auf dieser Auswahl wird nun das Usability Testing vorbereitet. Als übliches Verfahren haben sich dabei folgende Planungsschritte (PS) bewährt (vgl. wiederum [May1999]).

PS 1: Für jeden Anwendungsfall wird entschieden, ob die Anforderungen hinsichtlich der Benutzbarkeit eher in die Gruppe „einfache Erlernbarkeit“ *(„ease-of-learning“)* oder „effiziente Benutzung“ *(„ease-of-use“)* fallen. Diese Entscheidung wird vor allem im Zusammenhang mit den Benutzerprofilen und der erwarteten Häufigkeit der Benutzung der Funktionalität getroffen (z.B. selten benutzt => eher „ease-of-learning“).

PS 2: Für jeden Anwendungsfall werden Usability Ziele (*„goals“*) mit dem Kunden gemeinsam erarbeitet. Bekannte Usability Ziele sind etwa Einfachheit, Fehlerfreiheit (eine Übersicht an Usability Zielen fondet sich in [May1999]).

PS 3: Für die im vorherigen Schritt festgelegten Ziele werden Leistungsmaße (*„measures“*) bestimmt, mit denen das Erreichen der Ziele festgestellt werden kann (etwa „Zahl der Fehleingaben“).

PS4: Definition von Grenzwerten (*„thresholds“*), die das Erreichen bzw. das Verfehlen einer Usability Anforderung messbar machen.

Dies wird mit der Beschreibung der Anwendungsfälle und der daraus ableitbaren Testumgebung zum „Usability Testing Plan“ zusammengefasst.

Die selektierten Anwendungsfälle werden nun basierend auf den Usability Anforderungen durch die Entwickler unter Anwendung der XP Paradigmen (vgl. [Wel2001]) umgesetzt (Schritt 10). Das Resultat wird dem Kunden präsentiert. Ziel dieser Präsentation ist aber nicht nur, den Kunden über den Entwicklungsstand in Kenntnis zu setzen. Vielmehr soll der Kunde die Verwendung der im Rahmen der aktuellen Iteration umgesetzten Anwendungsfälle erlernen (Schritt 11). Basierend auf dieser Schulung kann der Kunde den Prototypen anhand der im „Usability Testing Plan" definierten Testfälle verwenden. Dazu werden idealerweise die Umgebungsbedingungen des für diesen Anwendungsfall aufgenommenen Ausführungsortes nachgestellt. Bei den Tests werden zum einen die definierten Usability Leistungsmaße erhoben und mit den definierten Zielwerten verglichen (Schritt 12), um quantitative Aussagen über die Benutzbarkeit des Iterationsergebnisses treffen zu können. Zum anderen geben die Testpersonen eine qualitative Rückmeldung bezüglich der verwendeten Prototypen. Die Ergebnisse des Usability Testing sowie die Bewertungen der Kunden resultieren in Änderungswünschen bzw. ergeben neue Anwendungsfälle (wieder Schritt 6). Diese bilden gemeinsam mit den in der aktuellen Iteration nicht umgesetzten Anwendungsfällen den Start für die nächste Iteration. Für abgenommene Testfälle kommt nun das XP Paradigma des „refactoring" zum Einsatz (vgl. [Wel2001]).

Zusammenfassung und Ausblick

Das hier vorgestellte Verfahren konnte unter Laborbedingungen mit inzwischen 60 Studenten gestestet werden, die sowohl die Rolle des Auftraggebers als auch die Rolle des Kunden einnahmen. Hierbei wurden verschiedene methodische Elemente evaluiert. Der oben vorgestellte Prozess ist eine Zusammenstellung der erfolgreichsten Elemente.. Darüber hinaus wurde das Verfahren der „benutzerzentrierten Entwicklung von mobiler Unternehmenssoftware" in verschiedenen Projekten in unterschiedlichen Branchen (Vertrieb Maschinenbau, Service Maschinenbau, Vertrieb Verlag, Health-Care) angewendet. Das Verfahren führte zu Projektergebnissen, die von den industriellen Auftraggebern als ausgesprochen gut bewertet wurden. Allerdings ist zu sagen, dass in allen Fällen der Auftraggeber und die Entwickler auf eine bereits durch andere Projekte etablierte Geschäftsbeziehung zurückblicken konnten. Damit konnte ein Nachteil der hier skizzierten Methode kompensiert werden. Diese Nachteile zeigten sich insbesondere in der Akquisition von Projekten. So wurde deutlich, dass die hier vorgestellte Methode sehr schwer in ein für alle Seiten zufriedenstellendes Vertragswerk überführt werden kann. Der Verkauf von abstrakten Ressourcen („etwa Manntage") zur Lösung eines nur grob spezifizierten Problems welches im Rahmen des Projektes detailliert wird, bedingt ein ausgesprochenes Vertrauensverhältnis zwischen Auftraggeber und -nehmer. Die Anwendung eines benutzerzentrierten Vorgehens bei der Entwicklung von mobilen Unternehmenssoftwaresystemen durch die Integration von Extreme Programming und Usability Engineering erscheint dennoch als viel versprechender Ansatz. Aktuell werden Werkzeuge (etwa eine Portalsoftware) entwickelt, die den skizzierten Prozess bzw. einzelne Schritte unterstützen, um etwa den Dokumentationsaufwand weiter zu reduzieren.

Literaturverzeichnis

[Ali2001] Ali, Mir Farooq; Pérez-Quiñones, Manuel A.; Abrams, Manuel A. (2001): A Multi-Step Process for Generating Multi-Platform User Interfaces with UIML. Technical Report cs.HC/0111025. Computing Research Repository (CoRR); http://citeseer.ist.psu.edu/560772.html

[Ba2001] Balzert, Helmut (2001): Lehrbuch der Softwaretechnik, Band 1 - Software-Entwicklung. Heidelberg: Spektrum Akademischer Verlag

[BA2004] Beck, Kent;Andres, Dirk Andres (2004): Extreme Programming Explained. Embrace Change. 2nd Edition, Addison Wesley

[BG2003] Burmester, Michael; Görner, Claus (2003): Das Wesen benutzerzentrierten Gestaltens. Machate, Joachim; Burmester, Michael (Hrsg.): User Interface Tuning. Software & Support Verlag

[Coc2001] Cockburn, Alistair (2001): Agile Software Development. Addison-Wesley

[Coc2005] Cockburn, Alistair Cockburn (2005): Crystal clear: a human-powered methodology for small teams, Addison-Wesley

[DIN2000] DIN (2000): DIN EN ISO 13407 Benutzer-orientierte Gestaltung i

[DIN2002] DIN (2002): DIN EN ISO 9241 – Ergonomic Requirements for office work with visual display terminals. 2002

[GM2006] Jones, Matt; Marsden, Gary (2006): Mobile Interaction Design: John Wiley & Sons

[Han2003] Hansmann, Uwe; Merk, Lothar; Nicklous, Martin S. (2003); Stober, Thomas: Pervasiv Computing. Second Edition. Berlin: Springer-Verlag

[KR2002] Kalakota, Ravi; Robinson, Marcia (2002): M-Business: The Race to Mobility. New York: McGraw-Hill

[KG2004] Köhler, André; Gruhn, Volker (2004): Mobile Process Landslcaping am Beispiel von Vertriebsprozessen in der Assekuranz. Key Pousttchi, Klaus Turowski (Ed.): Mobile Economy - Transaktionen, Prozesse, Anwendungen und Dienste. Proceedings zum 4. Workshop Mobile Commerce, Augsburg, Deutschland, 03.02.2004 - 03.02.2004, S. 12-24. Buchreihe LNI, GI-Lecture Notes in Informatics.

[Kot1997] Kotter, P. John (1997): Leading Change, Ernst Reinhardt Verlag, München, 1997

[Lov2005] Love, Steve (2005): Understanding Mobile Human-Computer-Interaction. Elsevier

[May1999] Mayhew, Deborah J. (1999): The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design. Morgan Kaufmann;

[Nor1986] Norman, Donald A. (1986): Cognitive Engineering. Norman. Donald A; Draper, Stephen W. (Eds.): User Centered System Design. Lawrence Erlbaum Associates;

[Nor1988] Norman, Donald A. (1988): The Design of everyday things. MIT Press;

[Rit2003] Ritz, Thomas (2003): "Mobile CRM Systeme"; in: ZWF-Zeitschrift für wirtschaftlichen Fabrikbetrieb; 12/2003; S.699-702

[Sal2005] Salmre, Ivo (2005): Writing Mobile Code. Upper Saddle River, NJ: Addison-Wesley

[SAP2003] SAP (2003): SAP Style Guide for White-Collar Worker PDAs; http://www.sapdesignguild.org/resources/sg_White-CollarWorkerPDAs/index.htm

[SR2001] Sandor, Reicher (2001): CUIML: A language for generating multimodal human-computer interfaces, Proceedings of the European UIML conference, citeseer.ist.psu.edu/sandor01cuiml.html

[Sch2004] Schwaber, Ken (2004): Agile Project Management with Scrum. Microsoft Press

[SR2003] Stender, Michael; Ritz, Thomas (2003): Überbetriebliche Ad-hoc-Anwendungsintegration im Mobile Business. in: Wolfgang Uhr, Werner Esswein, Eric Schoop (Hrsg.): Wirtschaftinformatik 2003/Band 1; Tagungsband zur 6. Internationalen Tagung Wirtschaftsinformatik; Dresden

[Stu2005] Studio 7.5 (2005): Designing for Small Screens. AVA

[Wei2002] Weiss, Scott (2002): Handheld Usability. John Wiley & Sons

[Wel2001] Wells, Don (2001): Extreme Programming: A gentle introduction. http://www.extremeprogramming.org/