Benutzerzentrierte Anforderungsanalyse für die Produktlinien-Entwicklung mobiler Unternehmenssoftware

Sebastian Damm, Thomas Ritz, Jakob Strauch

m²c Lab
FH Aachen
Eupener Str. 70
52066 Aachen
{s.damm, ritz, strauch}@fh-aachen.de

Abstract: Bei der Entwicklung mobiler Unternehmenssoftware ist es sinnvoll Methoden und Werkzeuge einzubeziehen, die es erlauben den Nutzungskontext der mobilen Applikation stärker zu berücksichtigen. Zudem werden viele mobile Lösungen für Benutzer entwickelt, die keine große Affinität zu Softwaresystemen haben (z.B. Handwerker). Der Artikel stellt das Zusammenspiel mehrerer domänenspezifischer Sprachen vor, die für die benutzerzentrierte Anforderungsanalyse genutzt werden. Diese sind dazu geeignet auch individuelle Anforderungen an das mobile System zu erfassen. Des Weiteren wird gezeigt, wie diese individuellen Anforderungen mit Hilfe einer Produktlinienstrategie kosteneffizient umgesetzt werden können.

1 Einleitung

Mobile Unternehmensanwendungen unterliegen im Gegensatz zu ihren Desktop-Pendants einem stetig variierenden Nutzungskontext (Netzverfügbarkeit, Lichtverhältnisse, Arbeitssituationen, etc.). Es gibt Ansätze und Vorgehensmodelle, um diese Besonderheiten in allen Phasen des Produktlebenszyklus möglichst vollständig zu erfassen und zu berücksichtigen (vgl. [Ri07]). Darüber hinaus adressieren mobile Anwendungen in der Industrie häufig Mitarbeiter, die bisher von der Büroautomatisierung weitestgehend unberührt geblieben sind. Die Herausforderung besteht nun darin, Applikationen zu entwerfen, die den individuellen Nutzungskontext berücksichtigen. Es müssen Domänenexperten in den Entwurfsprozess einbezogen werden, welche bisher nicht mit solchen Prozessen (etwa Prozessanalyse) oder Technologien (PDA, Tablet) vertraut waren. Die benutzerzentrierte Entwicklung mobiler Unternehmenssoftware (vgl. [RE08]) bietet dazu zwar ein erprobtes Vorgehensmodell, jedoch fehlten bisher die passenden Hilfsmittel (Werkzeuge, Checklisten, Fragebögen, etc.) zur idealen Umsetzung. In diesem Beitrag wird ein modellbasierter Ansatz für eine benutzerzentrierte Anforderungsanalyse vorgestellt. Die Anforderungen und Restriktionen für mobile Lösungen werden zusammen mit den Stakeholdern anhand einfach verständlicher grafischer Modelle herausgearbeitet.

Die Kosten individueller Entwicklung können gerade für KMU eine große Hürde bei der Integration mobiler Lösungen im eigenen Unternehmen darstellen. Daher werden im Zuge des Forschungsprojektes „Mobile Patterns as a Service“[1] zusätzlich Ansätze aus dem Produktlinien Engineering[2] in das Vorgehen integriert. Der aktuelle Forschungsstand im Bereich der mobilen Softwareentwicklung, die eingesetzten Technologien sowie Software Produktlinien werden in den folgenden Abschnitten vorgestellt. Darauf aufbauend werden neu entwickelte Hilfsmittel anhand eines Beispiels vorgestellt.

2 Stand des Wissens

2.1 Besonderheiten in der Entwicklung mobiler Unternehmenssoftware

Der Begriff Unternehmenssoftware hat sich als unscharfe Zusammenfassung von betrieblichen Anwendungssystemen etabliert. Die Einbindung von Mitarbeitern, die nicht an einem festen Standort arbeiten, ist das Ziel von mobiler Unternehmenssoftware. Die nachfolgende Definition von mobiler Unternehmenssoftware (siehe [Ri03]) erweitert diese 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.“

Aus dieser Definition ergeben sich unmittelbar die folgenden vier Anforderungen an ein Vorgehen zur Entwicklung von mobilen Unternehmenssoftwarelösungen:

Lokalität und Kontext müssen bei der Spezifikation von Software berücksichtigt werden. Insbesondere die Änderung der Orte, an denen eine Applikation verwendet wird, bringt Veränderungen z.B. an der verfügbaren Netzwerkinfrastruktur oder der Umgebungssituation (Geräuschpegel, Licht …) mit sich.

[1] BMBF Förderprogramm „IngenieurNachwuchs“, Förderkennzeichen 17N1709
[2] Kurz PLE oder S-PLE

Die *Interaktion* mit einer Applikation ändert sich. Schenkt man stationären Anwendungen durchschnittlich eine lange Aufmerksamkeit, so wird das Arbeiten mit mobilen IT-Systemen häufig dadurch charakterisiert, dass es sich um kurze und häufig unterbrochene Tätigkeiten handelt (vgl. [Sa05]). Dabei ist zu beachten, dass in mobilen Geschäftsprozessen oftmals Menschen arbeiten, die von der Büroautomatisierung bisher gering betroffen waren und eine Affinität zu Computeranwendungen daher nur eingeschränkt gegeben ist.

Es steht eine Vielzahl an möglichen *Endgeräten* mit unterschiedlichster Ausstattung zur Verfügung (vgl. [Ha03]). Die Auswahl und Nutzung muss ein zentraler Bestandteil des Software Engineering-Prozesses sein.

Um diesen Anforderung gerecht zu werden, haben sich benutzerzentrierte Entwicklungsmethoden bewährt [No86]. Wichtigste Vertreter iterativer Software-Entwicklungmethoden [Co01] sind hier Extreme Programming [BA04], Chrystal Methodologies [Co05] oder auch Scrum [Sc04]. Im Rahmen von vorherigen Forschungsarbeiten wurde eine integrierte Methode zur Entwicklung mobiler Lösungen entwickelt.

2.2 Benutzerzentrierte Entwicklung mobiler Unternehmenssoftware

Die benutzerzentrierte Entwicklung ist die Kombination von Usability Engineering [Ma99] und Extreme Programming (XP) zu einer integrierten agilen Entwicklungsmethode. Eine ausführliche Beschreibung der Methode findet sich in [Ri07]. Das Vorgehen ist dabei iterativ und es wird in jeder Iteration der Kontakt mit dem zuständigen Stakeholder gesucht.

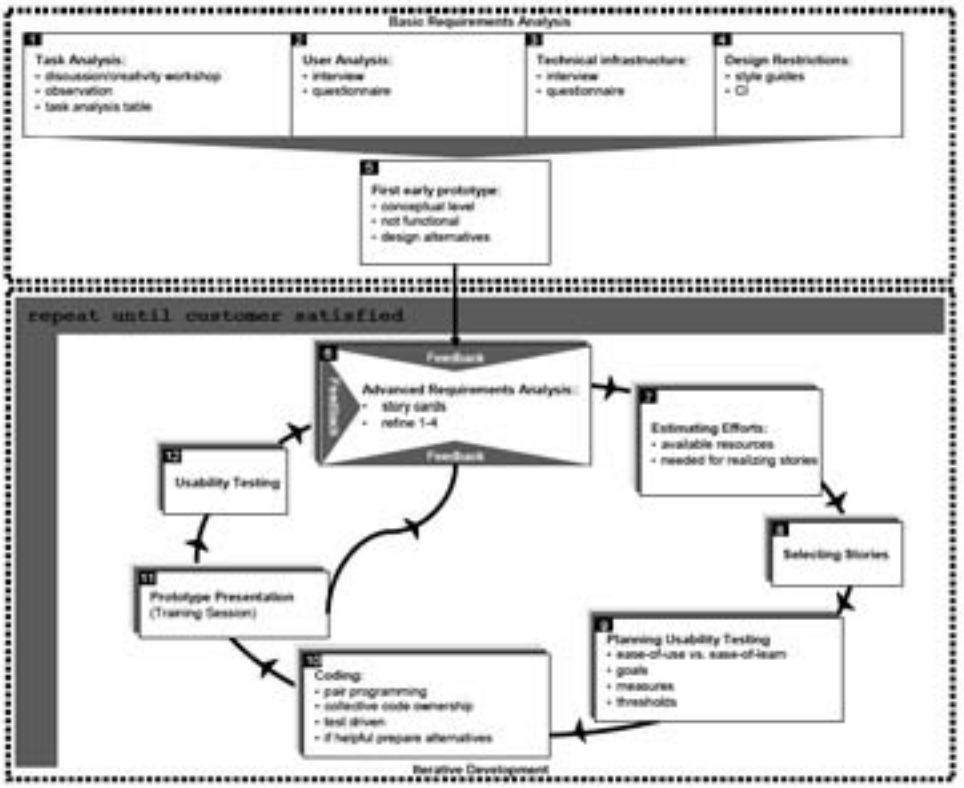

Abbildung 1: Benutzerzentrierte Entwicklung mobiler Unternehmenssoftware ([Ri07])

Die Anforderungsermittlung und die Entwicklung muss aus unterschiedlichsten Blickrichtungen (ergonomisch, technisch, etc.) betrachtet werden. Zudem müssen die Anforderungen gemeinsam mit den Stakeholdern ermittelt werden, sodass alle Beteiligten ein einheitliches Verständnis darüber haben. Das gewohnte Mittel dazu sind textuelle Beschreibungen oder Skizzen, die in Interviews festgehalten werden.

Erste Ansätze zur Berücksichtigung von Lokalitäten in der Anforderungsanalyse und damit verbundenen Infrastrukturfragen finden sich etwa im Mobile Process Landscaping (vgl. [KG04]). Es finden sich auch zahlreiche Varianten bzw. Erweiterungen der UML (etwa M-UML [SE03]), die mobile Aspekte einbeziehen. Dabei wird jedoch weniger die Modellierung von Anforderungen adressiert.

Durch den Einsatz grafischer domänenspezifischer Sprachen zur Visualisierung der speziellen Anforderungen und Restriktionen mobiler Software lässt sich das benutzerzentrierte Vorgehen verbessern.

2.3 Domänenspezifische Sprachen

Eine domänenspezifische Sprache[3] ist eine formale Sprache, die für ein abgegrenztes Problemfeld entwickelt wird und zu diesem Zweck eine möglichst problemnahe Notation verwendet. Neben textbasierten Sprachen können auch grafische Sprachen (Modelle) entworfen werden (vgl. [Co07]). Mit Hilfe grafischer DSLs können Problemaspekte visualisiert und in textbasierte Artefakte (Dokumente, XML, Programmcode, etc.) transformiert werden. Weit verbreitete Entwicklungsumgebungen wie eclipse[4] und Microsoft Visual Studio[5] bieten zu diesem Zweck entsprechende Frameworks an.

DSLs bieten eine Möglichkeit modellbasiert zu entwickeln. Eine Weitere bietet die UML-basierte „Model Driven Architecture“[6] (vgl. [SB07]). MDA definiert einen festen Satz an Modelltypen[7], wodurch der Entwicklungsprozess weitgehend vorgegeben ist. Mit dem DSL Ansatz hingegen kann man beliebige Modelltypen (Sprachen) definieren (vgl. [LW06]). Dadurch ist es möglich, Notationen zu verwenden, die nah an der Zieldomäne liegen. Abschnitt 3 beschreibt, wie DSLs eingesetzt werden, um die Ergebnisse einer Anforderungsanalyse den Beteiligten verständlicher darstellen zu können.

2.4 Software Produktlinien

Das Produktlinien Konzept hat seinen Ursprung in der klassischen Produktion von materiellen Gütern und wurde gegen Ende der 90er Jahre durch die Softwarebranche aufgegriffen. Das (Software) Produktlinien-Engineering[8] setzt auf „die organisierte *Wiederverwendung* und organisierte *Variabilität* auf Basis einer *gemeinsamen Plattform*“ (vgl. [BKP04]). Das PLE teilt den Entwicklungsprozess in zwei Phasen: Domänen- und Application-Engineering (vgl. [LRS07], [Po05]).

2.4.1 Domain Engineering

In der Phase des *Domain Engineerings* werden typischerweise folgende Schritte durchlaufen:

1. Eingrenzung der Zieldomäne
2. Festlegung der zu unterstützenden Plattform(en)
3. Entwicklung der Referenzarchitektur der Produktlinie

[3] engl. „Domain-Specific Language“, kurz DSL
[4] Eclipse Modeling Project
[5] Microsoft „DSL Tools“ und „Oslo“
[6] kurz MDA
[7] Die wichtigsten Modelle sind das plattformunabhängige Geschäftsprozessmodell (PIM) und das plattformspezifische Architekturmodell (PSM)
[8] Kurz (S)-PLE

4. Ermittlung der zu unterstützenden Anwendungsfälle, Funktionen und deren Variationsmöglichkeiten

Im zuletzt genannten Schritt ist es notwendig, auf Expertenwissen in der Zieldomäne zurückzugreifen.

2.4.2 Application Engineering

Die anfänglichen Mehrkosten, die das Domain Engineering mit sich bringt, sollen durch Vorteile wie einer kürzeren „Time-to-Market" aufgewogen werden (vgl. [LRS07]). Diese Vorteile entstehen dadurch, dass in der Phase des *Application Engineerings*[9] die wesentlichen Bestandteile eines Softwareproduktes bereits durch die Produktlinie vorkonfiguriert und generiert werden können.

Sowohl die Anwendungsfälle als auch die Varianten, die nicht von der Produktlinie abgedeckt werden, müssen individuell entwickelt werden. Die Erfahrungen und Ergebnisse solcher Individualentwicklungen können wieder in die Produktlinie zurückgeführt und diese somit für nachfolgende Produktableitungen genutzt werden. Eine gut realisierte Produktlinie kann die im vorigen Absatz genannten Mehrkosten durch eine hohe Wiederverwendbarkeit kompensieren.

Zusammengefasst kann man das Application Engineering in folgenden Schritten darstellen:

1. Anforderungsanalyse (z.B. Soll-Prozesse)
2. Abgleich mit den in der Produktlinie bereits vorhandenen Anwendungsfällen
3. Auswahl der Varianten, die den kundenindividuellen Anforderungen am besten entsprechen
4. Ggf. Implementierung der nicht abgedeckten Anforderungen
5. Ggf. Rückführung der Erfahrungen in die Produktlinie (Domain Engineering)

2.5 Zielsetzung

Aus den bisherigen Betrachtungen lassen sich u.a. folgende Fragestellungen ableiten:

1. Sind DSLs dazu geeignet die Anforderungsanalyse benutzerzentriert durchzuführen?
2. Lassen sich DSLs bei Produktlinien mit mobilem Anteil sinnvoll einsetzen?

[9] Bei der Entwicklung eines konkreten Produktes

3. Können Software Produktlinien die vielen individuellen Anforderungen mobiler Software aufnehmen und wiederverwendbar machen?

Zurzeit wird dies in der Domäne „Service im Anlagen- und Maschinenbau“ im Zuge des Forschungsprojektes bei einem Dienstleister mit Hilfe der entwickelten Werkzeuge evaluiert. Im Folgenden Auszug der noch laufenden Evaluation wird auf obige Fragestellungen eingegangen.

3 Beispiel

Für das nachfolgende Beispiel sei der Einfachheit halber vorausgesetzt, dass die Schritte 1-3 des Domain Engineerings bereits abgeschlossen sind. Das heißt, dass die Zieldomäne eingegrenzt wurde und Plattform sowie Architektur gewählt worden sind.

3.1 Domain Engineering: Prozesse und Varianten

Wie bereits erwähnt, benötigt man gute Kenntnisse in der Zieldomäne der Produktlinie, um diese effektiv planen zu können. Dieses Wissen kann anhand der Erfahrungen aus vergangenen Projekten gewonnen werden oder durch die Analyse der auf dem Markt vorhandenen Produkte in diesem Bereich. Die Kenntnisse des nachfolgenden Beispiels entstanden aus Diskussionen mit den Fachexperten, sowie der Analyse bestehender Applikationen. So wurden Anwendungsfälle festgehalten, die durch die Produktlinie unterstützt werden sollten. Ein solcher Anwendungsfall ist beispielsweise „Anlage identifizieren“.

Zur Prozess-Beschreibung wird eine DSL verwendet (Abbildung 2), die als Notationsgrundlage die BPMN[10] nutzt. Sie wurde um mobile Aspekte standardkonform erweitert bzw. eingeschränkt. So wurde festgelegt, dass „Pools“ einem Akteur zugewiesen werden. Um Verrichtungsorte im Prozessdiagram zu modellieren, wird der „Mobile Process Landscaping“ Ansatz von Gruhn et al (vgl. [KG04]) aufgegriffen. Die Methode beschreibt u.a. wie Prozessmodelle (EPK, UML-Aktivitätsdiagramme, etc.) um Lokationsgrenzen angereichert werden. Übertragen auf die verwendete BPMN werden dazu „Lanes“ als Lokationen konfiguriert und interpretiert. Dies spielt im Domain Engineering eine untergeordnete Rolle und wird im Application Engineering noch einmal aufgegriffen.

Ein Resultat des durchgeführten Domain Engineerings ist das Prozessmodell in Abbildung 2. Es hat im Wesentlichen zwei Ziele:

1. Prozessbeschreibung angereichert um Akteure und Lokationen durch den Softwareanalysten und den Domänenexperten

2. Festhalten der Variationspunkte und geplanten Varianten für die Softwareentwickler (vgl. [SP06])

[10] Business Process Modelling Notation, siehe http://www.bpmn.org

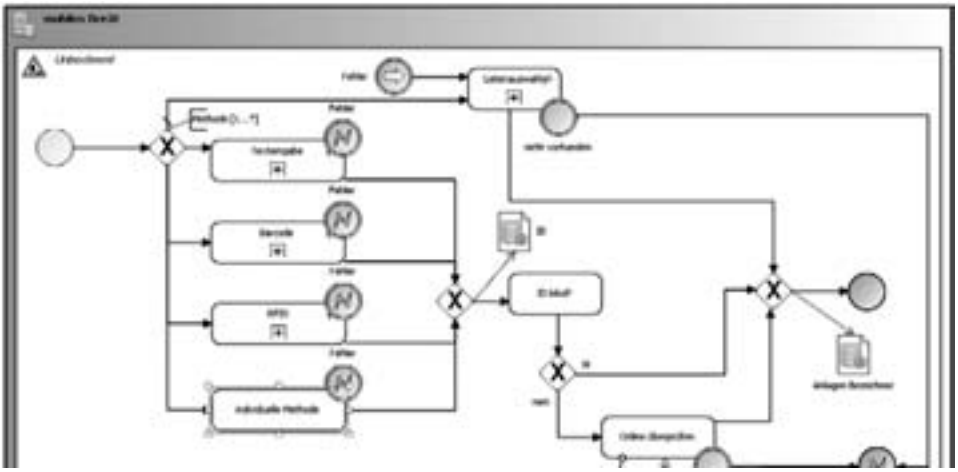

Abbildung 2: Prozessdiagram „Anlage identifizieren“ aus dem Domain Engineering

Die Abbildung beschreibt den detaillierten (System-)Prozess, der unterschiedliche Varianten für die Identifikationsmethode vorsieht (Texteingabe, Barcode, RFID,…). In einem konkreten Produkt wird später beim Application Engineering eine (oder mehrere) Methode(n) unter Berücksichtigung der produktspezifischen Anforderungen ausgewählt.

3.2 Application Engineering: Benutzerzentrierte Anforderungsanalyse

Nachfolgend wird die Vorgehensweise für das Application Engineering anhand der im Domain Engineering entwickelten Produktlinie vorgestellt. Diese beginnt mit der gemeinsamen Prozessanalyse durch den Softwareanalysten und dem (fiktiven) Kunden.

3.2.1 Grobe Prozessanalyse

Bei der Anforderungsermittlung zu einem konkreten Produkt wünscht der Kunde, dass der zukünftige Prozess mit Hilfe einer mobilen Lösung unterstützt werden soll. Abbildung 3 zeigt den Prozess, der mit dem Kunden zusammen definiert wurde.

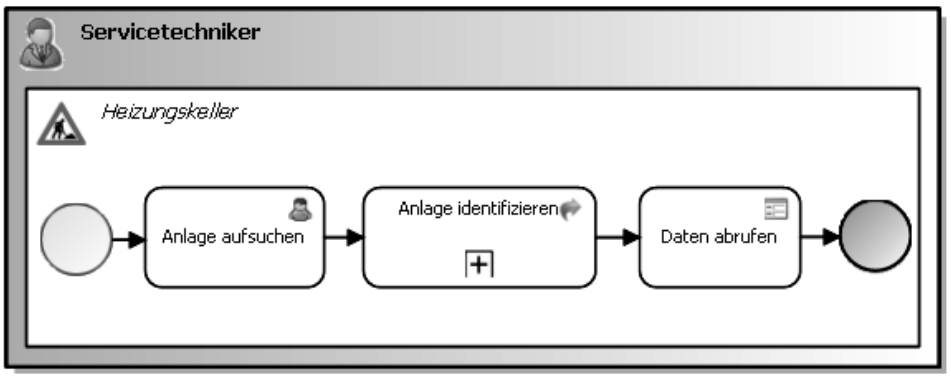

Abbildung 3: Gesamtprozess, u.a. mit Subprozess „Anlage identifizieren“

Es wird festgestellt, dass der abgebildete Subprozesses „Anlage identifizieren“ bereits durch die Produktlinie abgedeckt werden kann (siehe Application Engineering Schritt 1 und 2). Im nächsten Schritt muss herausgefunden werden, welche Varianten für den konkreten Anwendungsfall geeignet sind. Dabei ist besonders auf die Arbeitsumgebung der Servicetechniker des Kunden zu achten.

3.2.2 Lokationsprofil

Abbildung 4 zeigt einen Ausschnitt einer grafischen DSL, die eigens für die benutzerzentrierte Anforderungsanalyse entwickelt wurde. Sie dient zur Konfiguration typischer Arbeitsbedingungen von Nutzern einer mobilen Software-Lösung. Die Konfiguration aus dem Beispiel zeigt einen Heizungskeller, wie ihn ein Servicetechniker häufig vorfindet. Es wurden ein Foto der Lokation erstellt und Einzelheiten der Umgebung mit dem Monteur vor Ort ermittelt. Die ermittelten Restriktionen und Infrastruktur der Arbeitsumgebung wurden anschließend per „drag-and-drop“ auf dem Umgebungsbild platziert. Aus diesem visuellen Modell wird für den Softwareentwickler eine XML Konfigurations-Datei automatisch generiert.

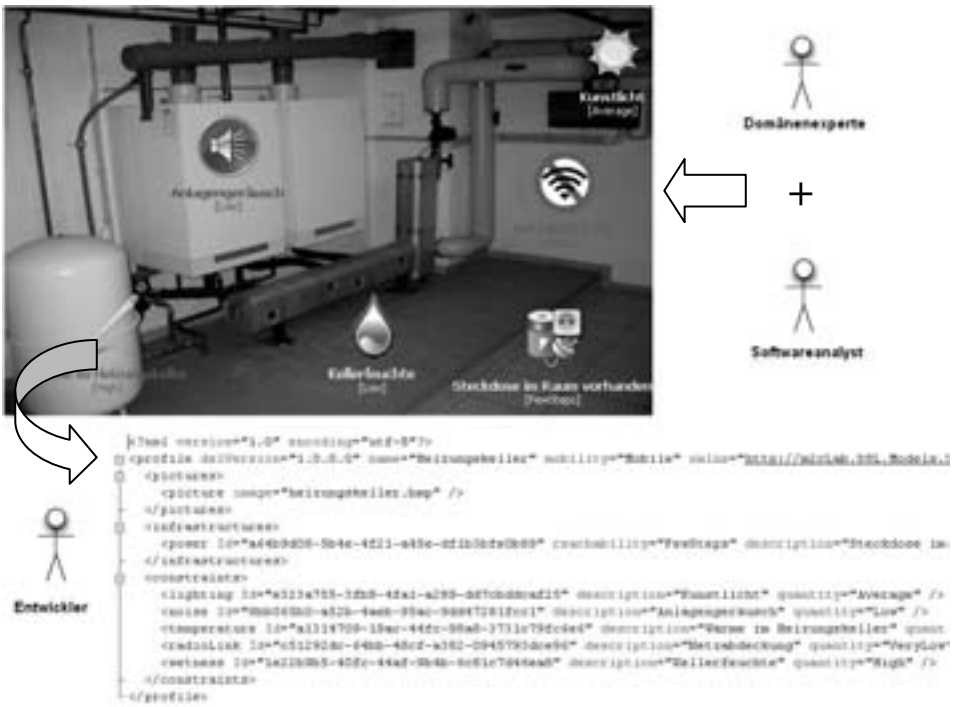

Abbildung 4: Schema zur Erstellung des „Mobile Location Profiles“

Es sei nochmal darauf hingewiesen, dass diese DSL eine formale Sprache ist. Somit ist die Abbildung eine gültige Instanz der zugrunde liegenden (domänenspezifischen) Sprache. Die Interaktion und die grafischen Elemente wurden formal definiert, auch wenn die Abbildung suggeriert, es handle sich nur um ein Bild mit grafischen Icons. Jedes Icon dient dem Zweck einen bestimmten Konfigurationseintrag zu generieren. Diese Konfiguration wird im nächsten Schritt die notwendigen Entscheidungshilfen liefern, um aus den unterschiedlichen Varianten eines Produktlinienartefaktes (z.B. Eingabemasken, Farbschemata oder Hardwaresensoren) die passende Variante auszuwählen.

3.2.3 Produktableitung

Die „Location Lane“ aus Abbildung 3 wird mit dem konkreten „Mobile Location Profile“ aus Abbildung 4 verknüpft. Das Ziel dabei ist es, eine Bewertung für die in Frage kommenden Varianten zu erstellen. Voraussetzung dafür ist, dass Regeln in Abbildung 2 für die einzelnen Varianten hinterlegt sind.

Die Auswertung der Regeln hat ergeben, dass die in der Produktlinie vorhandenen Varianten nicht optimal zum geplanten Produkt passen. So ist der Einsatz einer RFID Lösung aufgrund der Kellerfeuchte nicht zu empfehlen. Da die anvisierte Hardware nicht über einen Barcodescanner verfügt, entscheidet sich der Kunde für eine individuelle Identifikationsmethode. Es sollen mit der integrierten Kamera des Gerätes erfassbare QR-Codes genutzt werden. Abbildung 5 zeigt exemplarisch den Ablauf für diese neue Variante.

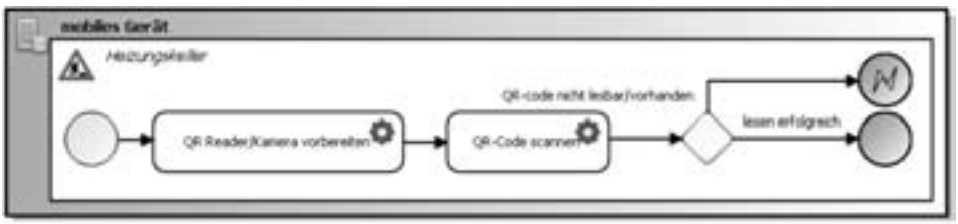

Abbildung 5: Detaillierter Subprozess „QR Code Scan“

Nach Abschluss des Projektes entschließt sich das Softwareunternehmen, diese neue Variante in die Produktlinie zu integrieren.

3.3 Ergebnis der Produktableitung

Der Kunde behält es sich desweiteren vor, die Anlage über eine Listenauswahl (Servicekunde -> Wartungsort -> Maschine) manuell auswählen zu können. Die in Abbildung 6 gewonnene Produktspezifikation wird anschließend verwendet, um die Implementierung und Anpassung des Produktes durchzuführen, die sich auf bereits existierende Komponenten der Produktlinie stützt und die oben beschriebene Individualentwicklung berücksichtigt. Das Ergebnis ist eine mögliche Instanz des Prozessdiagrammes aus Abbildung 2.

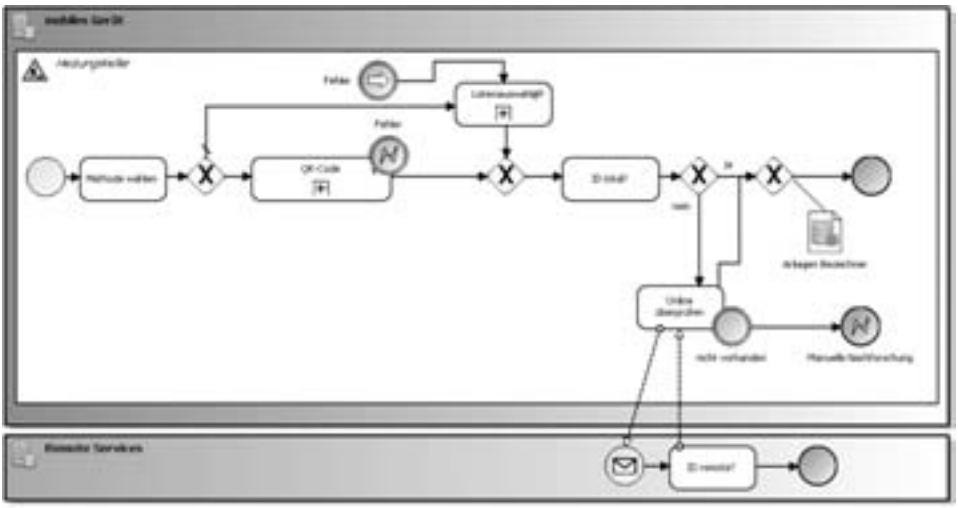

Abbildung 6: Aus der Produktableitung gewonnener Prozess

4 Fazit und Ausblick

Im Gegensatz zur klassischen Anforderungsermittlung, in der eine textuelle Liste der Anforderungen und Restriktionen mit den Kunden erstellt wird, kann der Entwickler diese nun mit den Stakeholdern (Auftraggeber, Servicetechniker, etc.) grafisch beschreiben. Dies ist für den Kunden transparenter und ohne Kenntnisse im Softwareengineering unmittelbar verständlich.

Das „Mobile Location Profile“ hilft dem gesamten Entwicklerteam eine bessere Vorstellung der Arbeitsumgebung des Heizungsmonteurs zu erlangen. Die Modelle mit den Photos und den Icons sagen auf einem Blick mehr als ein mehrere Seiten langes Anforderungsdokument[11]. Gleichzeitig wird auch eine Datenbasis zur Vorbereitung von Entwurfsentscheidungen der Hard- und Softwarespezifikation aufgebaut. So könnte es im gezeigten Beispiel sein, dass keine Datenverbindung via Funk am Verrichtungsort möglich ist, was einen online Datenabgleich unmöglich macht. Eine vorhandene Steckdose versetzt den Servicetechniker in die Lage den Geräteakku aufladen zu können, sofern erforderlich. Das Kunstlicht könnte die Farbdarstellung des Gerätes verfälschen, so dass beispielsweise Statusicons nicht unterscheidbar wären. Etwaige akustische Feedbacks der zu entwickelnden Anwendung könnten im Umgebungsgeräusch untergehen. All diese Informationen stehen nun auch in einem zentralen XML Repository zur Verfügung (vergl. Abbildung 4).

[11] Was nicht implizieren soll, dass derartige Dokumente dadurch überflüssig werden

Das Beispiel zeigt, wie sich wichtige Entwurfsentscheidungen auf den explizit modellierten mobilen Kontext stützen. Im Gegensatz zu klassischen Software-Engineering Methoden wurden die Anforderungen in einer für die Stakeholder unmittelbar verständlichen Form festgehalten. Darüber hinaus wurde das Modell wiederverwendet, um im Rahmen einer Produktableitung die passenden Varianten selektieren zu können. Die hier dargestellten Modelle wurden in der Entwicklungsumgebung Microsoft Visual Studio 2008 integriert.

Damit konnte gezeigt werden, dass DSLs und entsprechende Werkzeuge helfen können, mobile Unternehmenssoftware benutzerzentriert zu entwickeln. Des Weiteren können diese Ansätze verwendet werden, um basierend auf einer Produktlinienarchitektur eine konkrete Variante der Software nach individuellen Anforderungen der Nutzer abzuleiten. Weitere Evaluationen im Feld sollen zukünftig diese ersten vielversprechenden Resultate stützen.

Literaturverzeichnis

[BKP04] Böckle, G.; Knauber, P.; Pohl, K.: Software-Produktlinien. Methoden, Einführung und Praxis. dpunkt-Verl., Heidelberg, 2004.

[BA04] Beck, Kent;Andres, Andres, Dirk (2004): Extreme Programming Explained. Embrace Change. 2nd Edition, Addison Wesley

[Co01] Cockburn, Alistair (2001): Agile Software Development. Addison-Wesley

[Co05] Cockburn, Alistair Cockburn (2005): Crystal clear: a human-powered methodology for small teams, Addison-Wesley

[Co07] Cook, S.: Domain-specific development with Visual Studio DSL tools. Addison-Wesley, Upper Saddle River, NJ, 2007.

[Ha03] Hansmann, Uwe; Merk, Lothar; Nicklous, Martin S., Stober, Thomas (2003): Pervasiv Computing. Second Edition. Berlin: Springer-Verlag.

[KG04] Köhler, A.; Gruhn, V.: Mobile Process Landscaping am Beispiel von Vertriebsprozessen in der Assekuranz. Workshop Mobile Commerce, Universität Augsburg, 2.-3. Februar 2004. GI, 2004; S. 12–24.

[LW06] Lenz, G.; Wienands, C.: Practical Software Factories in NET. Gunther Lenz Christoph Wienands, Berkeley, CA, 2006.

[LRS07] Linden, F.; Rommes, E.; Schmid, K.: Software product lines in action. The best industrial practice in product line engineering. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, 2007.

[Ma99] Mayhew, Deborah J. (1999): The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design. Morgan Kaufmann;

[No86] Norman, Donald A. (1986): Cognitive Engineering. Norman. Donald A; Draper, Stephen W. (Eds.): User Centered System Design. Lawrence Erlbaum Associates;

[Po05] Pohl, K.; Böckle, G.; Linden, F.: Software product line engineering. Foundations, principles, and techniques ; with 10 tables. Springer, Berlin, 2005.

[RE08] Ritz, T.; Ellerweg, R.: Benutzerzentrierte Entwicklung mobiler Unternehmenssoftware. Teil 2 Die Iterative Entwicklung. In dot.net-magazin, 2008.

[Ri03] Ritz, Thomas (2003): Mobile CRM Systeme; in: ZWF-Zeitschrift für wirtschaftlichen Fabrikbetrieb; 12/2003; S.699-702

[Ri07] Ritz, T.: Die benutzerzentrierte Entwicklung mobiler Unternehmenssoftware. In (Gesellschaft für Informatik Hrsg.): MMS 2007: Mobilität und mobile Informationssysteme. 2nd conference of GI-Fachgruppe MMS, 2007.

[Sa05] Salmre, Ivo (2005): Writing Mobile Code. Upper Saddle River, NJ:Addison-Wesley.

[SE03] Saleh, Kassem; El-Morr, Christo (2003): M-UML: an extension to UML for the modeling of agent-based software systems. Information and Software Technology 2003

[Sc04] Schwaber, Ken (2004): Agile Project Management with Scrum. Microsoft Press

[SP06] Schnieders, A.; Puhlmann, F.: Variability Mechanisms in E-Business Process Families. In (Abramowicz, W.; Mayr, H. Hrsg.): 9th International Conference on Business Information Systems. BIS 2006, Klagenfurt, Austria, 2006.

[SB07] Stahl, T.; Bettin, J.: Modellgetriebene Softwareentwicklung. Techniken, Engineering, Management. dpunkt-Verl., Heidelberg, 2007.