Ein Ansatz zur modellbasierten Entwicklung mobiler Benutzungsschnittstellen mit aufgabenorientierter Visualisierung

Peter Forbrig, Georg Fuchs, Daniel Reichart, Heidrun Schumann

Universität Rostock
Fakultät für Informatik und Elektrotechnik
Albert-Einstein-Str.21
18051 Rostock
Peter.Forbrig@informatik.uni-rostock.de
Georg.Fuchs@informatik.uni-rostock.de
Danel.Reichart@informatik.uni-rostock.de
Heidrun.Schumann@informatik.uni-rostock.de

Abstract: Die Entwicklung von Nutzungsschnittstellen für mobile Geräte stellt neue Herausforderungen an Softwareentwickler. Durch den Entwurf von Aufgaben- und Dialogmodellen können Nutzungsschnittstellen für unterschiedliche Geräte spezifiziert werden. Konkrete Geräte und Kontextsituationen erfordern auch die Nutzung alternativer Interaktions- und Präsentationstechniken. Wir stellen hier einen modellbasierten Ansatz vor, der eine Visualisierung, angepasst zu den jeweiligen Aufgaben, unterstütz.

1 Einleitung

Die zunehmende Verbreitung mobiler Geräte ermöglicht eine steigende Anzahl von Anwendungen. Durch die große Menge an unterschiedlichen Geräten werden Entwickler interaktiver Software allerdings vor eine Reihe von Herausforderungen gestellt. Um nicht für jedes Gerät eine eigene Nutzungsschnittstelle entwickeln und pflegen zu müssen, ist es sinnvoll, Modellspezifikationen zu nutzen, die plattformunabhängig sind. In diesem Beitrag wollen wir unsere Herangehensweise am Beispiel der Entwicklung einer Software zur Unterstützung von Instandhaltungsmaßnahmen darstellen. Dieses Szenario wird derzeit im Rahmen des 4. Landesforschungsschwerpunktes IuK „Multimediales Content Management in mobilen Umgebungen mit multimodalen Nutzungsschnittstellen“ bearbeitet. Ein Ziel besteht darin, Wartungsingenieuren auf mobilen Geräten Zugriff auf Instandsetzungsanleitungen zu geben.

2 Modellbasierte Entwicklung interaktiver Systeme

Unser Entwicklungsprozess beginnt mit der Spezifikation von Aufgabenmodellen. Diese enthalten alle vom System und dessen Nutzern abzuarbeitenden Aufgaben und deren Beziehungen untereinander. Es ist möglich, zeitliche Abhängigkeiten, Alternativen, optionale Teilaufgaben und Iterationen zu definieren. Neben der sequenziellen Iteration, bei der ein Schritt beendet werden muss, bevor der nächste angefangen werden darf, haben wir auch einen Operator für Instanziteration eingeführt. Dieser bewirkt, dass mehrere Instanzen eines Aufgabentyps parallel abgearbeitet werden können.

Das Ergebnis der Aufgabenmodellierung ist ein Aufgabenbaum, der alle Interaktionsmöglichkeiten zusammen mit den benutzten Werkzeugen und den zu bearbeitenden Arbeitsgegenständen strukturiert beschreibt. Zur Entwicklung eines Dialogmodells gibt es bereits verschiedene Ansätze. Beispielsweise wird im TERESA-Projekt [BCM04] das Dialogmodell automatisch generiert, indem aus dem Aufgabenmodell so genannte Enabled Task Sets gewonnen werden, welche direkt in das Dialogmodell eingehen. Das bedeutet, die Nutzungsschnittstelle passt sich nach der Abarbeitung jeder Teilaufgabe automatisch an den neuen Zustand des Aufgabenmodells an. Gegen diese Methode sprechen in unserem Fall zwei Dinge: Erstens wird durch die Zulassung von Instanziterationen die Menge der Enabled Task Sets unendlich groß, da zu einem konkreten Zeitpunkt eine beliebige Anzahl von Instanzen einer Aufgabe aktiv sein kann. Zweitens kann diese Vorgehensweise für mobile Geräte aufgrund ihrer geringen Displaygröße bei komplexen Aufgabenmodellen zu einer stark überladenen Nutzeroberfläche führen. Wir haben uns letztlich dafür entschieden, das Konzept der abstrakten Dialoggraphen [SE96] zur Spezifikation der Dialogstruktur zu nutzen. Wir bieten damit eine Alternative Variante zur Vorgehensweise bei TERESA an, deren Nutzung dem Anwender überlassen werden kann. Ein Dialoggraph ist ein gerichteter Graph, dessen Knoten, die Dialogsichten, einzelne Dialoge der Nutzungsschnittstelle repräsentieren. Es gibt verschiedene Typen von Dialogsichten, die sich unterschiedlich verhalten. Beispielsweise verhindert ein modaler Dialog das Aktivieren jedes anderen, gerade sichtbaren Dialogs, während eine multiple Dialogsicht das Erzeugen mehrerer Instanzen dieses Dialoges erlaubt. Dialogsichten sind durch so genannte Transitionen miteinander verbunden, welche mögliche Dialogübergänge darstellen. Der Typ der Transition entscheidet zudem, ob der Ausgangsdialog bei diesem Übergang offen bleibt oder geschlossen wird. Mehr Details dazu findet man dazu auch in [FWD06].

Die Aufgaben aus dem Aufgabenmodell werden dabei den einzelnen Dialogsichten aus dem Dialoggraphen zugeordnet. Jeder Dialog dient der Bearbeitung einer oder mehrerer Aufgaben und es kann festgelegt werden, welche Aufgaben Dialogübergänge nach sich ziehen. Durch diesen Ansatz lassen sich bereits sehr präzise die Gesamtstruktur der Nutzungsschnittstelle und mögliche anvisierten Interaktionsfolgen spezifizieren. Durch die Gruppierung der Aufgaben in Dialogsichten werden Aufgabenmodelle häufig eingeschränkt. So kann z.B. durch die sequentielle Anordnung von Dialogen die Reihenfolge nebenläufiger Teilaufgaben erzwungen werden. Gerade für Geräte mit geringer Displaygröße können solche Einschränkungen Sinn machen, um die Anzahl möglicher Interaktionen und damit auch den Bedarf an Platz in der graphischen Nutzungsschnittstelle zu begrenzen.

Der letzte Schritt in unserem Entwicklungsprozess ist die Verfeinerung der Dialoge bis auf Komponentenebene. Es werden jeder Aufgabe Interaktionskomponenten zugeordnet und anschließend das Gesamtlayout des Dialogs festgelegt. Diese Komponenten können auf verschiedenen Abstraktionsebenen liegen und unterschiedlich komplex sein. So können bereits vorliegende komplexe oder sehr spezielle Komponenten, mit generischen, plattformunabhängigen Komponenten kombiniert werden. Diese Zuordnung von Komponenten zu Aufgaben kann auch bei einer Umstrukturierung des Dialoggraphen erhalten bleiben, so dass bei einer Änderung der Anforderungen einmal getroffene Designentscheidungen wieder berücksichtigt werden können.

Basierend auf der Metapher, dass Aufgaben Arbeitsgegenstände und Werkzeuge zugeordnet sind, können aus den Modellen erste Klassendiagramme extrahiert werden. Die Aufgaben werden dabei zu den Methoden der zugeordneten Arbeitsgegenstände. Eine Transformation durch Design Patterns kann dann zu Entwurfsmodellen führen. Das soll in diesem Papier aber nicht diskutiert werden. Hier wollen wir uns auf die Erzeugung der Benutzungsoberfläche konzentrieren. Dabei spielen Klassendiagramme allerdings auch eine Rolle. Nach unserer Philosophie wird ein Navigationsdialog aus Aufgabenmodellen und abgeleiteten Dialogspezifikation erzeugt. In den Bearbeitungsdialog fließen dann aus Objektmodellen generierte XUL-Spezifikationen mit ein. Abbildung 1 stellt diesen Sachverhalt grafisch dar. Dieser Sachverhalt wird hier aber nicht diskutiert.

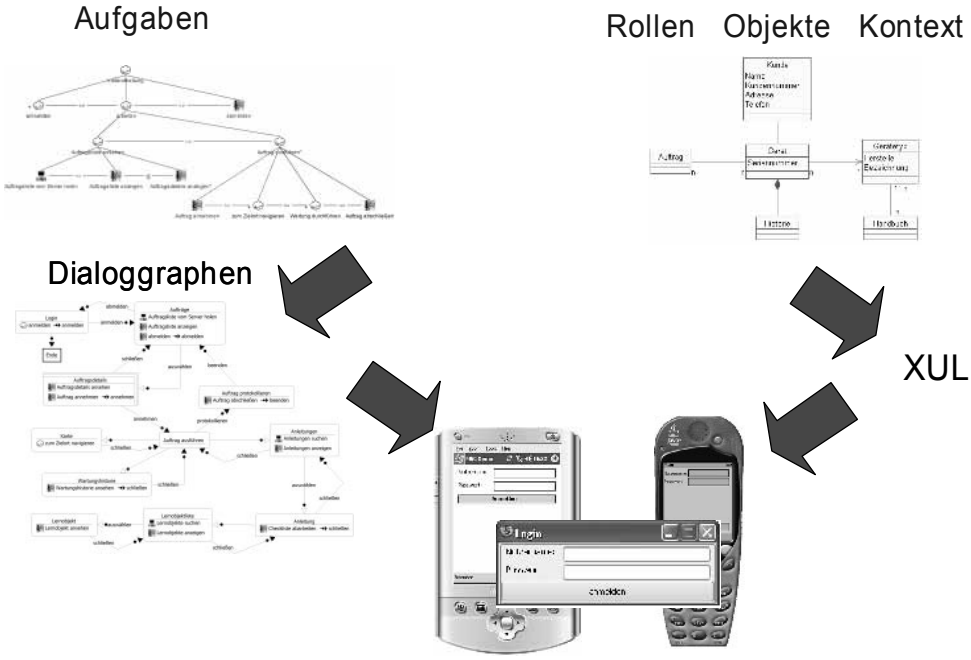

Abbildung 1: Modelle für Navigations- (links) und Bearbeitungsdialog (rechts)

Ausgangspunkt der Entwicklung bildeten ursprünglich Aufgabenmodelle in der Form, wie sie von CTTE [CTT06] bekannt sind. Es zeigte sich aber schnell, dass deren Ausdrucksform nicht stark genug war. Iterative und optionale Aufgaben reichten nicht aus, um die Probleme der Praxis korrekt zu modellieren. Aus diesem Grunde wurde die Instanziteration für Aufgabenmodelle eingeführt. Sie erlaubt es, eine neue Iteration bereits zu beginnen, bevor die vorhergehende Iteration beendet ist.

2.1 Modellierungswerkzeuge

Zur Unterstützung des oben beschriebenen Prozesses wurden von uns Werkzeuge entwickelt, welche die Modellierung und Simulation der oben genannten Modelle ermöglichen und diese anschließend weiterverarbeiten können, z.B. durch die Generierung von Quellcode für eine oder mehrere Zielplattformen. Diese Tools basieren auf dem Eclipse Modeling Framework (EMF) [EMF06] und fügen sich als Plug-Ins in die Eclipse DIE [Ec06] ein.

Ausgangspunkt für die Entwicklung modellbasierter Werkzeuge unter EMF ist die Spezifikation von Metamodellen. In unsrem Falle wird dazu UML und Rational Rose genutzt. Im Rahmen des Projektes wurden Metamodelle für Aufgabenmodelle, abstrakte Dialoggraphen, Objektmodelle und Nutzermodelle entwickelt und daraus mit Hilfe von EMF ein ganzes Paket von Editoren in Form von Eclipse-Plugins generiert. Diese haben einen sehr generischen Charakter, eignen sich aber gut, um hierarchische Modelle, beispielsweise Aufgabenmodelle, zu bearbeiten. Modelle werden von EMF im XMI-Format abgelegt. Die generierten textbasierten XML-Editoren sind für die Entwicklung der Werkzeuge nützlich, einen Anwender aber nicht zumutbar. Auf ihrer Basis können aber mit Hilfe des Graphical Editing Framework [GEF06] komfortablere, graphische Editoren entwickelt werden. Eines dieser graphischen Werkzeuge ist der Dialoggrapheditor. Er ermöglicht die Erstellung und Bearbeitung kompletter Dialoggraphen und die Zuordnung von Aufgaben zu einzelnen Dialogsichten. Außerdem wird hier definiert, ob nach Abarbeitung einer Aufgabe eine bestimmte Transition ausgelöst werden soll.

Neben Bearbeitungsfunktionen unterstützt unsere Toolsammlung auch die Simulation von Dialoggraphen unter Berücksichtigung der Simulation von Aufgabenmodellen. Damit wird es dem Entwickler ermöglicht, bereits frühzeitig das Verhalten der Modelle mit seiner Vorstellung oder der des zukünftigen Anwenders zu vergleichen. Zu einem Aufgabenmodell wird üblicher Weise für jede Rolle ein Dialoggraph erstellt. Diese werden gleichzeitig animiert. Abbildung 1 zeigt eine prototypisch generierte Benutzungsoberfläche als Resultat der Animation eines Dialoggraphen. Im linken Teil von Abbildung 2 wird das animierte Aufgabenmodell präsentiert. In der Mitte sieht man die prototypische Benutzungsoberfläche, die zunächst nur aus Buttons besteht. Im rechten Teil sieht man alle spezifizierten Modelle und im unteren Teil des Bildschirmes befinden sich Informationen zum abgearbeiteten Szenario. Der Zustand des gleichfalls sichtbaren animierten Aufgabenmodells ist ebenfalls sichtbar. Das animierte Aufgabenmodell ist dafür verantwortlich, dass „Maßnahme zur Abarbeitung wählen“ im aktuellen Zustand nicht aktivierbar ist.

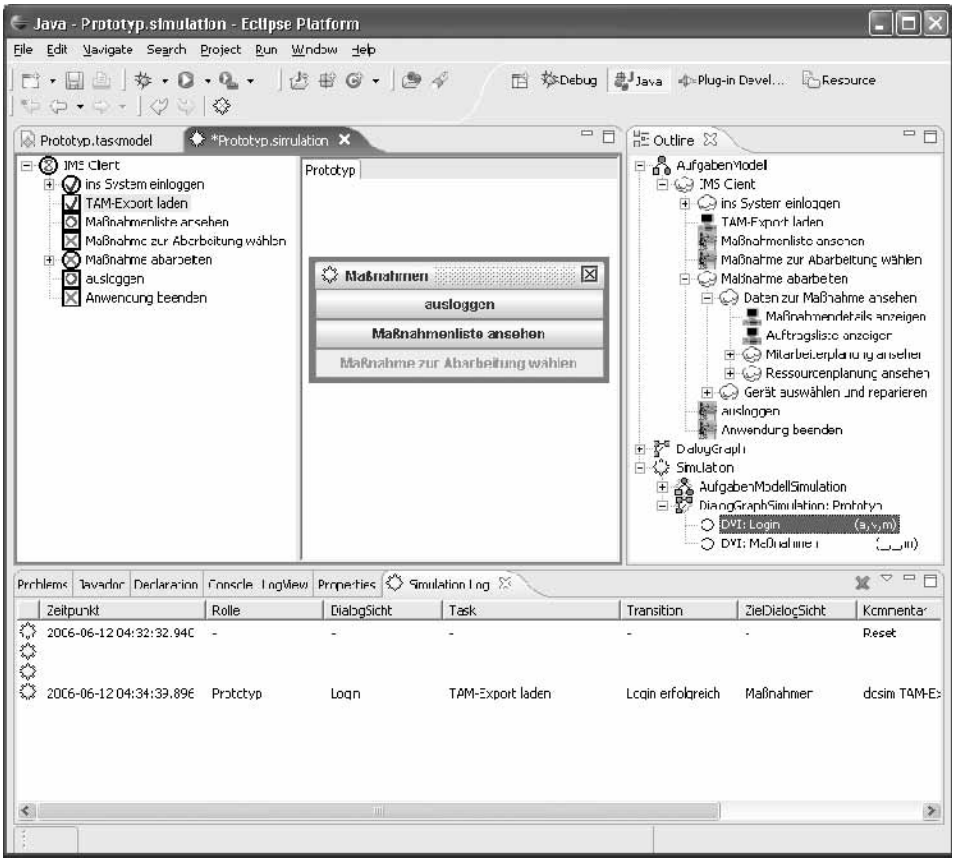

Abbildung 2: Animierter Prototyp einer Anwendung

2.2 Spezifikation der Benutzungsoberfläche

Als Beschreibungssprache für die Benutzungsoberfläche wird XUL [XUL06] verwendet. Mit Hilfe eines selbst entwickelten XUL-Editors [DFR05] können die generierten Dialogbeschreibungen anschließend verfeinert werden, ohne dass Beziehungen zu den Ausgangsmodellen verloren gehen. Damit können nachfolgende Animationen bereits auf Basis dieser Spezifikationen animiert werden. Dies leistet wiederum ein in die Animation integrierter XUL-Interpreter, der für Java und .Net vorliegt. Für J2ME existiert ein Compiler.

Im Anschluss an die Modellierung kann aus den Modellinformationen unter Anwendung von Templates auch Quellcode für einen Prototypen generiert werden, der auf dem gewünschten Zielgerät lauffähig ist. Es existieren derzeit Templates zur Erzeugung von GUIs für Java/SWING und das .NET Compact Framework. In Zukunft kann durch die Einführung neuer Templates mit relativ geringem Entwicklungsaufwand auch der Wechsel zu anderen Zielplattformen erreicht werden.

3 Aufgabenorientierte Visualisierung

Für Instandhaltungsmaßnahmen ist es wichtig, dass der Wartungsingenieur auch visuell die wichtigsten Teile der aktuellen Maßnahme gut wahrnehmen kann. Dies ist gerade auf kleinen Displays eine große Herausforderung und geschieht in unserem Fall unter Anwendung von Focus&Context-Techniken. [Ke98]. Die Erstellung solcher Wartungsanleitungen wird durch ein Autorenwerkzeug unterstützt. Die Durchführung der Wartung wird durch Aufgabenmodelle spezifiziert. Der Aufgabenmodelleditor wurde dafür so erweitert, dass für jede Aufgabe zusätzliche Informationen zur Visualisierung hinterlegt werden können. Über den Mechanismus der Metamodelle war das relativ einfach möglich. Die graphischen Informationen an sich, wie Bilder und in ihnen enthaltene Features, werden in einem separaten Autorenwerkzeug erstellt, welches ebenfalls als Eclipse-Plug-In verfügbar ist. Als Ausgangsbilder werden Raster- und Vektorgraphiken unterstützt und die Definition der einzelnen Features erfolgt, wie in Abbildung 3 zu erkennen, durch Polygone. Eine detaillierte Diskussion findet man dazu in [FS06].

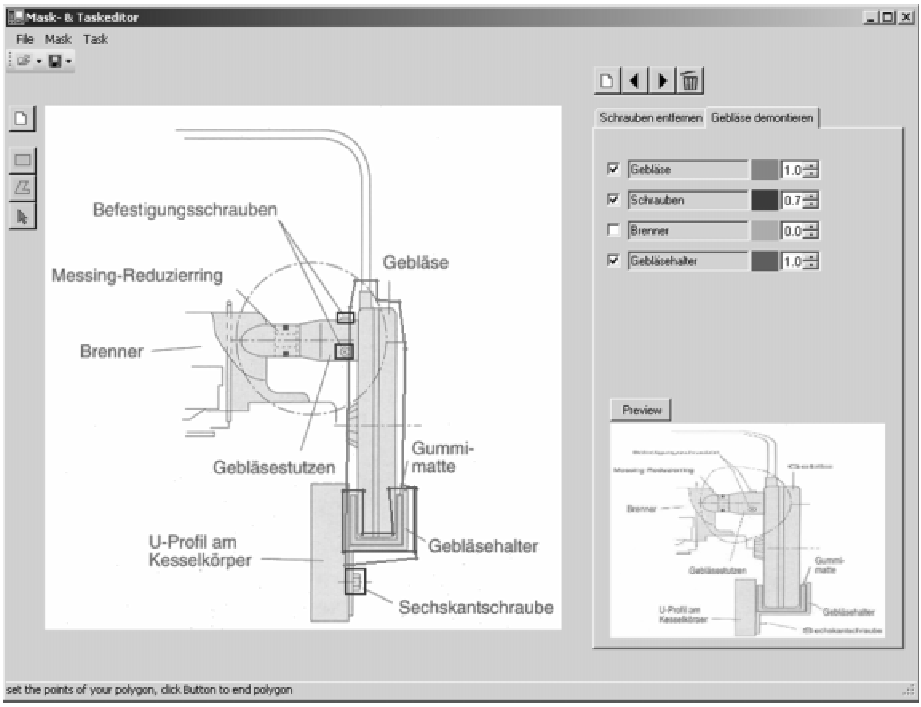

Abbildung 3: Autorenwerkzeug zur Erstellung von Visualisierungsinformationen

Durch die Zuordnung der unterschiedlichen Visualisierungen zu Aufgaben wird es später bei der Ausführung der Aufgabe möglich, de jeweilige Visualisierung mit ihren Interaktionen zu präsentieren.

Es hat sich gezeigt, dass die Spezifikation von Arbeitsabläufen über Aufgabenmodelle nicht immer auf die Gegenliebe der Softwareentwickler stößt.

Doch auch hier sollte eine Visualisierung erfolgen, die den Aufgaben angepasst ist. Aus diesem Grunde wurde ein Editor entwickelt, der Aufgabenmodelle in Form von Aktivitätsdiagrammen präsentiert. Dabei handelt es sich um einen strukturierten Editor, der nicht das Ziehen von Kanten vorsieht, sondern nur das Einfügen von Folgen, Alternativen und Schleifen erlaubt. Die Visualisierung erfolgt dann automatisch. Einen Eindruck vom Aussehen dieses Editors gibt Abbildung 4. Im rechten Teil des Bildes sieht man die Struktur des zugehörigen Aufgabenmodells. Bei der gerade erfolgten Animation wurde die Aufgabe „große Inspektion durchführen“ aktiviert. Diese Aufgabe wurde erfolgreich abgeschlossen, was durch ein grünes Häkchen angezeigt wird. Die anderen Alternativen wurden nicht ausgeführt, was durch ein graues x angezeigt wird.

Aus den im Editiermodus verfügbaren Operationen, sichtbar im linken oberen Teil des Fensters, ist noch einmal der strukturierte Charakter der Manipulationen ersichtlich. Die Position der Anwendung der Operation muss durch eine Selektion mit der Maus gewählt werden. Das System zeigt mögliche Positionen beim überqueren mit der Maus an.

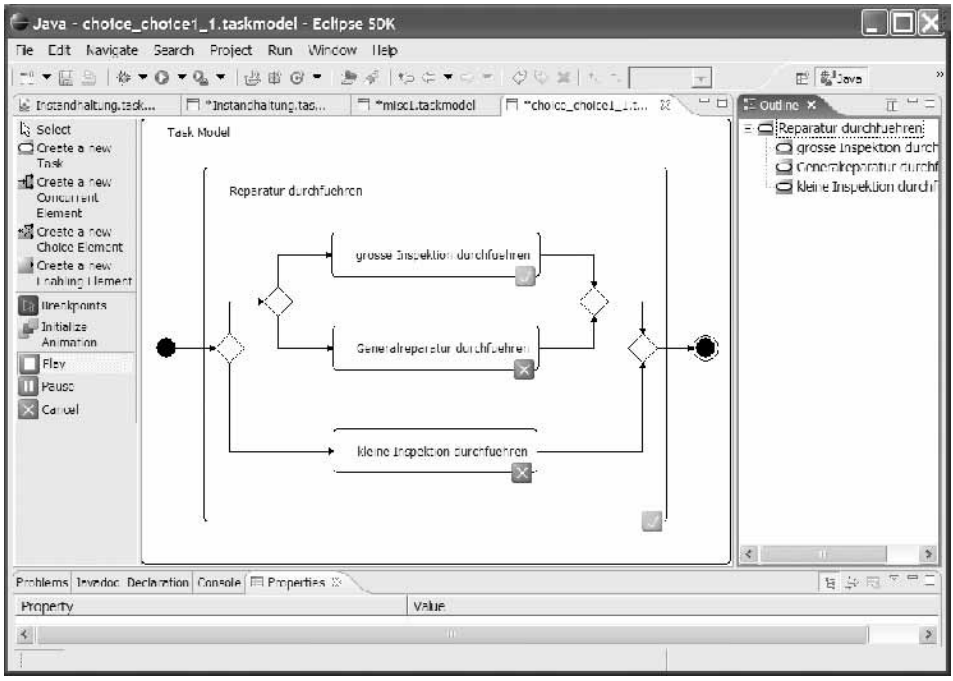

Abbildung 4: Animiertes Aktivitätsdiagramm

Aufgabenmodelle haben eine größere Ausdruckskraft als Aktivitätsdiagramme. So erlauben diese beispielsweise in ihrer Originalform nicht, dass eine Folgaktivität eine vorhergehende unterbricht, was bei Aufgabenmodelle durch die temporale Relation „Deaktivierung“ ausgedrückt wird. Dies kann im Aktivitätsdiagramm aber durch Stereotypen spezifiziert werden. Damit können die Aufgabenmodelle zu einer Erweiterung der Notation in den UML-Diagrammen beitragen.

4 Zusammenfassung und Ausblick

In den vorherigen Kapiteln haben wir eine mögliche Strategie zur Entwicklung von mobilen Nutzungsschnittstellen vorgestellt und erläutert. Für einige Schritte innerhalb dieses Entwicklungsprozesses wurden bereits Werkzeuge entwickelt, die eine bequeme Arbeitsweise ermöglichen. Es wurde angestrebt, untersichtliche Sichten in speziellen Notationen auf Modele zu ermöglichen. Das gilt speziell für das Aufgabenmodell, das konzeptionell um die Instanziteration erweitert wurde.

Eine der Herausforderungen für die Zukunft ist die Erzeugung kompletter Anwendungen aus Modellen. Durch die immer wiederkehrende Anwendung der Entwicklungsprozesse und die Analyse bestehender Nutzungsschnittstellen werden sich generische Teilmodelle in Form von Patterns herauskristallisieren, die zukünftig in den Entwicklungsprozess integrierbar sein müssen. Erste Gedanken findet man dazu in [WFD06].

Literaturverzeichnis

[BCM04] Berti, S.; Correani, F.; Mori, G.; Paternò, F.; Santoro, C. (2004): A transformation-based environment for designing multi-device interactive applications. In: Proceedings of the 9th international conference on Intelligent user interfaces. Funchal, Januar 2004.

[CTT06] CTTE: http://giove.cnuce.cnr.it/ctte.html, einges. 9.8.06.

[DFR 05] Dittmar, A.; Forbrig, P.; Reichart, D.; Wolff, A. (2005): Linking GUI Elements to Tasks – Supporting an Evolutionary Design Process. In: Proc. of TAMODIA 2005, Gdansk, September 2005.

[Ec06] Eclipse Homepage. http://www.eclipse.org/, einges.: 09.08.06.

[EMF06] Eclipse Modeling Framework Homepage. http://www.eclipse.org/emf/, einges.: 9.8.06.

[FWD06] Forbrig, Peter; Wolff, Andreas; Dittmar, Anke; Reichart, Daniel: Tool Support for an Evolutionary Design Process using XML and User-Interface Patterns. Proc. CUSEC 2006, Montreal, January, 2006.

[FRS05] Fuchs G.; Reichart, D.; Schumann, H.; Forbrig, P. (2006): Maintenance Support – Case Study for a Multimodal Mobile User Interface. IS&T/SPIE's 16th Annual Symposium Electronic Imaging: Multimedia on Mobile Devices II, 15.-19. January 2006, San Jose, California, USA.

[FS06] Fuchs, G., and Schumann, H.: Visualization in Multimodal User Interfaces of Mobile Applications. Proceedings IRMA'06, 17th International Conference of the Information Resources Managment Association, May 2006, Washington D.C., USA.

[GEF06] Graphical Editing Framework Homepage. http://www.eclipse.org/gef/

[Ke98] Keahey, T. A. (1998): The generalized detail-in-context problem. In: Proc. of the IEEE Symposium on Information Visualization, IEEE Visualization, Oktober 1998.

[SE96] Schlungbaum, E.; Elwert, T. (1996): Dialogue Graphs – A Formal and Visual Specification Technique for Dialogue Modelling. Springer, 1996.

[WFD06] Wolff, A.; Forbrig, P.; Dittmar, A.; Reichart, D.: Tool Support for an Evolutionary Design Process using Patterns, Proc. Of the Workshop: Multi-channel Adaptive Context-sensitive (MAC) Systems: Building Links Between Research Communities, Glasgow, 2006.

[XUL06] XUL: http://www.mozilla.org/projects/xul/, einges. 9.8.06.