A. M. Heinecke, H. Paul (Hrsg.): Mensch & Computer 2006: Mensch und Computer im Struktur*Wandel*.
München, Oldenbourg Verlag, 2006, S. 195-202

Modellbasierte Entwicklung mobiler multimodaler Nutzungsschnittstellen

Peter Forbrig, Georg Fuchs, Daniel Reichart, Heidrun Schumann

Universität Rostock, Institut für Informatik

Zusammenfassung

Die Entwicklung von Nutzungsschnittstellen für mobile Geräte stellt neue Herausforderungen an Softwareentwickler. Durch den Entwurf von Aufgaben- und Dialogmodellen können geräteübergreifend Nutzungsschnittstellen spezifiziert werden. Konkrete Geräte und Kontextsituationen erfordern auch die Nutzung alternativer Interaktions- und Präsentationstechniken. Wir stellen hier einen Ansatz vor, der die Integration von Sprache und Visualisierung am Beispiel einer Anwendung im Bereich Instandhaltungsmanagement vorantreibt.

1 Einleitung

Unser tägliches Leben wird mehr und mehr von den Möglichkeiten der mobilen Kommunikation beeinflusst. Die zunehmende Ausbreitung drahtloser Netzwerke ermöglicht eine Vielzahl von Anwendungen sowohl im privaten als auch im geschäftlichen Bereich. Durch die große Menge an unterschiedlichen Geräten werden Entwickler interaktiver Software vor neue Herausforderungen gestellt. Um nicht für jedes Gerät eine eigene Nutzungsschnittstelle entwickeln und pflegen zu müssen, verfolgen wir einen Ansatz, bei dem plattformunabhängige Modelle eine zentrale Rolle spielen.

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. Landesforschungsschwerpunkt IuK „Multimediales Content Management in mobilen Umgebungen mit multimodalen Nutzungsschnittstellen“ in Zusammenarbeit mit der SIV.AG bearbeitet. Das derzeitige Instandhaltungsmanagementsystem erlaubt es, Protokolle mit den anstehenden Arbeitspaketen auszudrucken und dem Wartungstechniker mitzugeben. Vor Ort kann dieser Reparaturzeiten, Messergebnisse, eingebaute Ersatzteile und andere Rückmeldeinformationen dort eintragen. Später müssen die im Protokoll eingetragenen Daten wieder ins System eingepflegt werden.

Ein Ziel ist es, dieses Protokoll durch die Entwicklung einer mobilen Komponente zu ersetzen und so beiden Seiten einen Wettbewerbsvorteil zu verschaffen: Die Verwaltung profitiert

durch den geringeren Pflegeaufwand und aktuellere Daten, während der Wartungstechniker, insbesondere bei der Reparatur älterer oder selten eingesetzter Anlagen, auf technische Dokumentationen zugreifen und sich so vor Ort Hilfe holen kann.

2 Modellbasierte Entwicklung

Unser Entwicklungsprozess beginnt mit der Spezifikation von Aufgabenmodellen. Diese enthalten alle vom System und dessen Nutzern abzuarbeitenden Aufgaben und deren Beziehungen untereinander. Es ist u.a. 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 strukturiert beschreibt.

Zur Entwicklung eines Dialogmodells gibt es bereits verschiedene Ansätze. Beispielsweise wird im TERESA-Projekt (Berti et al. 2004) das Dialogmodell automatisch generiert, indem aus dem Aufgabenmodell sogenannte 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 wäre durch die Einführung der Instanziteration die Menge der Enabled Task Sets unendlich groß, da zu einem konkreten Zeitpunkt eine beliebige Anzahl an 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 (Schlungbaum & Elwert 1996) zur Spezifikation der Dialogstruktur zu nutzen.

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 sogenannte Transitionen miteinander verbunden, welche mögliche Dialogübergänge darstellen. Der Typ der Transition entscheidet zudem, ob der Ausgangsdialog bei diesem Übergang offen bleibt oder geschlossen wird.

Wir nutzen eine Kombination der beiden obigen Konzepte zur Spezifikation der Nutzungsschnittstelle. Die Aufgaben aus dem Aufgabenmodell werden dabei den einzelnen Dialogsichten aus dem abstrakten 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, wie z.B. ein Browserfenster oder ein Videoplayer, 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. Abbildung 1 zeigt noch einmal eine Übersicht über unseren gesamten Entwicklungsprozess.

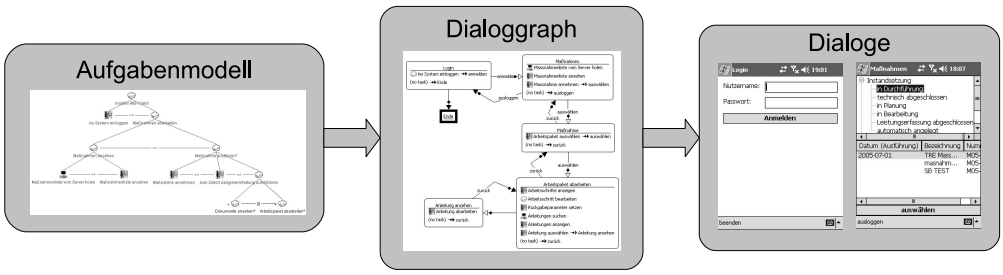

Abbildung 1: Entwicklungsprozess für Nutzungsschnittstellen

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) und fügen sich als Plug-Ins in die Eclipse IDE ein. Somit kann ein Softwareentwickler direkt innerhalb der Entwicklungsumgebung damit arbeiten.

Ausgangspunkt für die Entwicklung modellbasierter Werkzeuge unter EMF ist die Spezifikation von Metamodellen, im Normalfall mittels UML. Wir haben unter anderem 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 Plugins wurden von uns anschließend teilweise nachbearbeitet und dienen als Basis für komfortablere, graphische Werkzeuge.

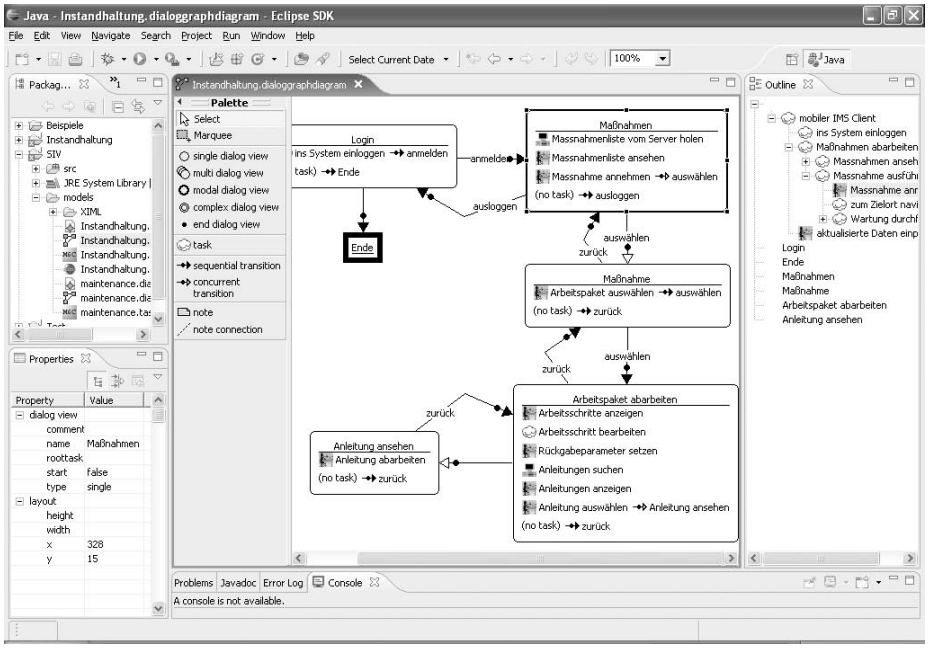

Abbildung 2: Dialoggrapheditor

Eines dieser graphischen Werkzeuge ist der Dialoggrapheditor (siehe Abbildung 2). Er basiert auf dem Graphical Editing Framework (GEF) und 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 Aufgabenmodellen und Dialoggraphen. Damit wird es dem Entwickler ermöglicht, bereits frühzeitig das Verhalten der Modelle mit seiner Vorstellung oder der des zukünftigen Anwenders zu vergleichen.

Im Anschluss an die Modellierung kann aus den Modellinformationen unter Anwendung von Templates 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.

Als Dialogbeschreibungssprache verwenden wir XUL. Mit Hilfe eines von uns entwickelten XUL-Editors (Dittmar et al. 2005) können die generierten Dialogbeschreibungen anschließend verfeinert werden, ohne dass Beziehungen zu den Ausgangsmodellen verloren gehen. Wir haben XUL-Interpreter für Java und .NET und einen XUL-Compiler für J2ME entwickelt, so dass die Dialogbeschreibungen auf unterschiedlichen Plattformen verwendet werden können. Zudem kann der Dialogdesigner bereits während der Simulationszeit eines Dia-

loggraphen XUL zur Darstellung der Dialoginhalte verwenden und so schon einen sehr konkreten Eindruck der späteren Nutzungsschnittstelle erhalten.

3 Szenario „mobiles Instandhaltungsmanagement“

Unsere Werkzeuge kommen derzeit bei der Entwicklung einer Software zur Unterstützung von Technikern bei Instandhaltungsmaßnahmen zum Einsatz. Dieser Software liegt ein Framework zugrunde, welches unter anderem das Interpretieren XML-basierter Dialogbeschreibungen zur Laufzeit realisiert. Diese beinhalten neben XUL auch XAML, z.B. zur Einbindung eigener Komponenten für Visualisierungsaufgaben, und SALT (Speech Application Language Tags) zur Spezifikation von Sprachinteraktionen. Ein Teil der Instandhaltungsmanagement-Software beschäftigt sich mit der Präsentation von Wartungsanleitungen und soll hier kurz vorgestellt werden.

In Abhängigkeit von der Art der Instandhaltungsmaßnahme und dem Typ der zu wartenden Anlage wird dem Techniker auf seinem PDA eine Anleitung zur Verfügung gestellt, mit deren Hilfe er Schritt für Schritt eine Aufgabe abarbeitet. Da er hierfür in vielen Fällen beide Hände frei haben und sich auf seine Arbeit konzentrieren muss, sind andere Interaktionstechniken wie die Ein- und Ausgabe per Sprache notwendig. Weiterhin ist es sinnvoll, technische Sachverhalte visuell darzustellen und relevante Teile der Visualisierung hervorzuheben. Dies ist gerade auf kleinen Displays eine große Herausforderung und geschieht in unserem Fall unter Anwendung von Focus&Context-Techniken (Keahey 1998).

Abbildung 3: Digitale Wartungsanleitung auf einem PDA

Abbildung 3 zeigt einen Screenshot der Wartungsanleitung. Zum aktuellen Arbeitsschritt passend werden dem Techniker eine generelle Beschreibung als Text angezeigt und Detailinformationen vorgelesen. In einer Visualisierungskomponente wird parallel dazu der technische Sachverhalt graphisch dargestellt. Hierzu werden einzelne Bildausschnitte, sogenannte Features, hervorgehoben. Dies kann zum einen durch Einfärbung geschehen, gleichzeitig kann ein Feature aber auch in Abhängigkeit von seiner Wichtigkeit innerhalb der aktuellen Aufgabe vergrößert werden, während andere Bildausschnitte verkleinert dargestellt werden und so in den Hintergrund rücken. Durch diese Technik können Grafiken mehrfach wiederverwendet werden, nur die Wichtigkeiten der Features unterscheiden sich je nach Instruktion.

Im Hintergrund sendet das System zusätzlich eine Suchanfrage an einen eLearning-Server und wertet dessen Antwort aus, um bei Vorhandensein weiterer Informationen zum jeweiligen Arbeitsschritt dem Nutzer Zugriff auf diese anzubieten. Das Glühlämpchen in der rechten unteren Ecke der Visualisierungskomponente zeigt in diesem Fall das Vorhandensein von eLearning-Objekten an.

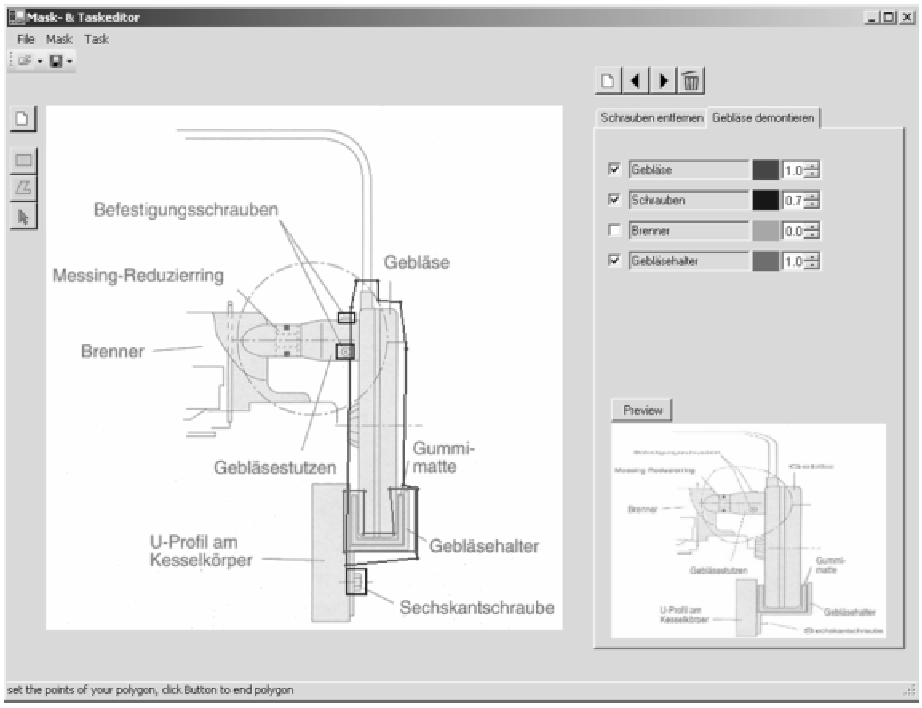

Abbildung 4: Autorenwerkzeug zur Erstellung von Visualisierungsinformationen

Die Visualisierungskomponente ermöglicht weiterhin das Vergrößern und Verschieben des Bildausschnittes und die gesonderte Selektion und Hervorhebung einzelner Features. Diese und andere Interaktionen können per Stift oder Spracheingabe geschehen. Nachdem der Nutzer den Arbeitsschritt beendet hat, kann er, wieder per Sprache oder konventioneller

Eingabe, zum nächsten Schritt wechseln. Damit ist eine durchgehend freihändige Bedienung des Gerätes denkbar.

Auch die Erstellung solcher Wartungsanleitungen wird von uns durch Werkzeuge unterstützt. Zur Spezifikation einer Anleitung bieten sich wiederum Aufgabenmodelle an. Damit wird es beispielsweise möglich, Anleitungen für eine ganze Klasse von zu wartenden Anlagen in einem einzigen Modell unterzubringen und zur Laufzeit die im konkreten Fall überflüssigen Teilaufgaben herauszufiltern. Zu diesem Zweck haben wir den Aufgabenmodelleditor so erweitert, dass für jede Aufgabe zusätzliche Informationen zur Sprachinteraktion und zur Visualisierung hinterlegt werden können. Die graphischen Informationen an sich, wie Bilder und in ihnen enthaltene Features, werden in einem separaten Autorenwerkzeug erstellt, welches ebenfalls als Eclipse-Plugin verfügbar ist. Als Ausgangsbilder werden Raster- und Vektorgraphiken unterstützt und die Definition der einzelnen Features erfolgt, wie in Abbildung 4 zu erkennen, durch Polygone.

4 Ausblick

In den vorherigen Kapiteln haben wir gezeigt, wie unsere Strategie zur Entwicklung von mobilen Nutzungsschnittstellen aussieht und diese an einem Beispiel näher erläutert. Für viele Schritte innerhalb dieses Entwicklungsprozesses bieten wir Werkzeugunterstützung an, diese ist aber bei weitem noch nicht lückenlos. Eine der Herausforderungen für die Zukunft ist sicherlich die Erzeugung kompletter Dialogbeschreibungen aus unseren Modellen. Hier gilt es, Kompromisse zu finden zwischen einer vollkommen automatischen Generierung und der Bereitstellung von Möglichkeiten, manuell Designentscheidungen treffen zu können. Durch die immer wiederkehrende Anwendung unserer Entwicklungsprozesse und die Analyse bestehender Nutzungsschnittstellen werden sich Patterns herauskristallisieren, die in Form von Werkzeugunterstützung in den Prozess mit eingehen werden.

Literaturverzeichnis

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.

Biehl, N.; Düsterhöft, A.(2005): Speech Control for Mobile Devices in Maintenance Support Scenarios. IEEE Int. Conference on Natural Language Processing and Knowledge Engineering (NLP-KE 2005), 30. Oktober-1. November 2005, Wuhan, China.

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.

Eclipse Homepage. http://www.eclipse.org/.

Eclipse Modeling Framework Homepage. http://www.eclipse.org/emf/.

Forbrig, P.; Dittmar, A.; Reichart, D.; Sinnig, D. (2004): From Models to Interactive Systems – Tool Support and XIML. In: Proc. Workshop #4 Making Model Based User Interface Practical, IUI/CADUI 2004, Funchal, Januar 2004.

Forbrig, P.; Schumann, H. (2005): Advanved Multi-Modal User Interfaces – Visualisation, Natural Language and Platform Independence. Workshop: Mobile Computing and Ambient Intelligence: The Challenge of Multimedia, Dagstuhl, May 2005.

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. Januar 2006, San Jose, California, USA.

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

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

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

Kontaktinformationen

Peter Forbrig – Lehrstuhl Softwaretechnik;
Email: peter.forbrig@informatik.uni-rostock.de, Tel.: 0381 498 7620

Georg Fuchs – Lehrstuhl Computergraphik;
Email: georg.fuchs@informatik.uni-rostock.de, Tel.: 0381 498 7484

Daniel Reichart – Lehrstuhl Softwaretechnik;
Email: daniel.reichart@informatik.uni-rostock.de, Tel.: 0381 498 7623

Heidrun Schumann – Lehrstuhl Computergraphik;
Email: heidrun.schumann@informatik.uni-rostock.de, Tel.: 0381 498 7490

Universität Rostock
Institut für Informatik
Albert-Einstein-Straße 21
18051 Rostock