EMODE – ein Ansatz zur werkzeugunterstützten Modellierung multimodaler, adaptiver Benutzerschnittstellen

Matthias Winkler[1], Matthias Heinrich[1], Alexander Behring[2], Joachim Steinmetz[2], Waltenegus Dargie[3]

[1]SAP Research CEC Dresden
Chemnitzer Str. 48
01187 Dresden
matthias.winkler@sap.com
matthias.heinrich@sap.com

[2]TU Darmstadt
FG Telekooperation
Hochschulstraße 10
64289 Darmstadt
behring@tk.informatik.tu-darmstadt.de
joachim@tk.informatik.tu-darmstadt.de

[3]TU Dresden
Lehrstuhl Rechnernetze
Nöthnitzer Str. 46
01187 Dresden
waltenegus.dargie@tu-dresden.de

Abstract: Gegenwärtig ist die Erstellung multimodaler, adaptiver Anwendungen (MMA) ein sehr zeitaufwändiger Prozess, bei dem Entwickler verschiedene Technologien miteinander kombinieren müssen. Die Unterstützung des Entwicklungsprozesses durch geeignete Werkzeuge ist bisher unzureichend. In diesem Artikel wird die im Rahmen des EMODE-Projekts entwickelte Methodik zur Entwicklung von MMA beschrieben. Des Weiteren wird eine integrierte Toolchain präsentiert, welche die Methodik realisiert. Ziel von EMODE ist die Verbesserung der Effizienz der Entwicklung von MMA.

1 Einführung

Für Nutzer von Anwendungen ist die Interaktion mit dem IT System oft nicht Hauptbestandteil der Arbeit. Vielmehr interagieren sie mit der Anwendung, während sie gleichzeitig einer anderen Hauptaktivität nachgehen. Die Wahl eines passenden Interaktionsmediums, wie dies bei multimodalen, adaptiven Anwendungen möglich ist, kann dem Nutzer mehr Flexibilität und Freiheit bieten.

Jedoch ist die Entwicklung von MMA ein kosten- und zeitaufwändiger Prozess. Ursache ist u.a. die Komplexität interaktiver Systeme sowie das Fehlen standardisierter Methoden und integrierter Werkzeuge, die den Entwicklungsprozess unterstützen. Auch die Integration der verschiedenen Aspekte der Anwendungsentwicklung, insbesondere der Benutzerschnittstelle und der fachlichen Logik, bietet Potenzial zu Verbesserung.

Es existieren verschiedene Ansätze zur Entwicklung multimodaler, adaptiver Anwendungen (z.B. [CAL02, LIM04, ST04]). Dabei wird jedoch der Fokus weniger auf die Methodik gerichtet, sondern eher Aspekte der Integration und der Darstellung der multimodalen Benutzerschnittstelle beleuchtet.

EMODE [EMO05] stellt eine Methodik auf Basis der Model Driven Architecture (MDA) [MDA07] bereit. Dabei liegt der Fokus auf der engeren Integration der einzelnen Teilnehmer am Entwicklungsprozess, sowie der Entwicklungs- und Laufzeitumgebung. Ziel ist die Verbesserung der Effizienz des Entwicklungsprozesses von MMA.

Im Folgenden werden die EMODE-Methodik, das Metamodell und die Transformationen beschrieben. Anschließend wird eine Toolchain vorgestellt, die auf der EMODE-Methodik basiert. Schließlich werden diese Punkte anhand eines Beispiels verdeutlicht.

2 Der EMODE-Ansatz

Es wird zunächst die EMODE-Methodik erläutert, welche durch ein Metamodell, Transformationen und die EMODE-Toolchain umgesetzt wird. Des Weiteren wird die Methodik anhand eines Beispiels verdeutlicht.

2.1 Methodik

Die Erstellung von MMA gestaltet sich schwierig, da sehr unterschiedliche Aspekte einer Anwendung beachtet werden müssen, die von verschiedenen Stakeholdern vertreten werden. Solche Aspekte sind z.B. die Aufgaben des Anwenders in der realen Welt und deren Umsetzung durch das System, sowie Interaktionsmöglichkeiten des Anwenders mit der Anwendung. Die EMODE-Methodik schenkt diesen Aspekten besondere Beachtung. Sie setzt sich aus dem Projektteam, einer Reihe von Entwicklungsphasen, Artefakten und einer konzeptionellen Architektur zusammen.

Das *Projektteam* besteht aus Endanwendern, Anwendungsentwicklern und UI Designern. Die intensive Einbeziehung von Endanwendern in den Entwicklungsprozess ist besonders wichtig, um eine bessere Qualität der Anwendung im Hinblick auf deren Eignung zur Unterstützung des Anwenders bei seinen Aufgaben zu erreichen. Die Integration des Endanwenders in den Entwicklungsprozess erfolgt dadurch, dass seine Anforderungen an die Anwendung als formale Beschreibung den Ausgangspunkt der EMODE-Modellierung bilden. Des Weiteren wird durch den EMODE-Entwicklungsprozess die schnelle Entwicklung von Prototypen unterstützt, sowie das Testen dieser gemeinsam mit dem Anwender. Das Feedback kann dann in den iterativen Entwicklungsprozess einfließen. Anwendungsentwickler und UI Designer sind hierbei für die Entwicklung der entsprechenden Teile einer Anwendung verantwortlich. Sie bringen spezifisches Wissen ihrer Domäne in den Entwicklungsprozess ein.

Die EMODE-Methodik beschreibt vier verschiedene *Entwicklungsphasen*, in denen unterschiedliche Modelle erstellt werden (vgl. Abbildung 1). Während der *Requirements Analysis Phase* werden die Anforderungen des Anwenders an die Anwendung gesammelt. Auf dieser Basis werden in der *High-level Designphase* das Ineinandergreifen von Nutzer- und Systemaufgaben mit Hilfe eines *Task-Modells*, sowie Konzepte der Anwendung modelliert. Das Task-Modell kann weiter verfeinert werden, bis der Ablauf der Anwendung detailliert beschrieben ist. Des Weiteren ist die Erweiterung um Kontexter-

eignisse und –informationen möglich. Zu Beginn der *Detailed Design Phase* wird das Task-Modell in das *Abstract User Interface* (AUI)-Modell und das *Functional Core Adapter* (FCA)-Modell transformiert. In diesen Modellen findet eine detaillierte Beschreibung des User Interface und der Schnittstellen zur Anwendungslogik statt. Die Modellierung des User Interface wird von UI Designern weitergeführt. Die Geschäftslogik, die von den Anwendungsentwicklern erstellt wird, wird über den FCA angebunden. Das Task-Modell ist hier zentraler Bestandteil, welcher den gesamten Ablauf der Anwendung aus Sicht des Anwenders widerspiegelt und gleichzeitig AUI- und FCA-Modelle in Beziehung setzt. Änderungen am Task-Modell können Änderungen am AUI-Modell und FCA-Modell zur Folge haben. Diese Kopplung der Modelle verbessert die Konsistenz von UI und Funktionalität: die Interessen der Stakeholder werden aufgeteilt aber gleichzeitig über klare Schnittstellen in enge Verbindung gesetzt. Weitere Modelle zur Beschreibung von Kontext- und Geräteinformationen im Hinblick auf unterstützte Ein- und Ausgabemodalitäten werden auch in dieser dritten Phase erstellt. In der letzten Phase (*Implementation*) wird schließlich der Code der Anwendung durch eine Modell-zu-Code-Transformation erstellt. Abbildung 1 fasst die Phasen, Artefakte und Transformationen der EMODE-Methodik nochmals zusammen.

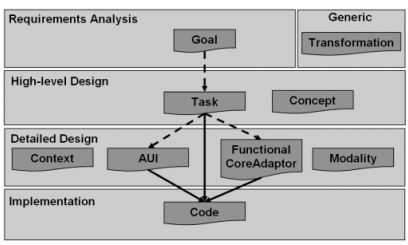

Abbildung 1: Phasen, Artefakte und Transformationen

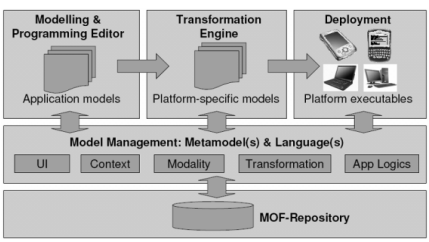

Abbildung 2: Konzeptionelle Architektur

Die konzeptionelle Architektur in Abbildung 2 beschreibt das Zusammenspiel verschiedener Komponenten, die Funktionalitäten zur Unterstützung der einzelnen Entwicklungsphasen zur Verfügung stellen. Wichtige Funktionalitäten sind das Modellmanagement, Modellierungs-, Programmier- und Transformationsunterstützung. Die konzeptionelle Architektur wurde als Toolchain implementiert (siehe Abschnitt 2.3).

Die EMODE-Methodik bildet die Grundlage für eine effizientere Entwicklung multimodaler, adaptiver Anwendungen. Anwendungsentwickler, UI Designer und Anwender haben die Möglichkeit, in einem iterativen Entwicklungsprozess effizient zusammenzuarbeiten. Das wird durch die enge Verknüpfung der Kernmodelle Task, AUI und FCA unterstützt.

2.2 Metamodell und Transformationen

Die EMODE-Methodik wird mit Hilfe des Prinzips Modell-getriebener Software-Entwicklung umgesetzt. Metamodell-Pakete definieren hierbei die verschiedenen Artefakte und ihre Semantik. Die verschiedenen Aspekte der Entwicklung (vgl. Abschnitt 2.1) können durch Abbildungen verknüpft und somit integriert werden. Transformationen werden für die Automatisierung von Entwicklungsschritten, sowie zur Wahrung der Konsistenz der Modelle untereinander genutzt.

Insbesondere drei Pakete (vgl. Abschnitt 2.1) spielen bei EMODE eine wesentliche Rolle bzw. heben sich von anderen Projekten (z.B. [CAL02, LIM04, ST04]) ab. Im *Task-Modell* wurden Konzepte aus ConcurTaskTrees (CTT) [PAT97] und UML Aktivitätsdiagrammen kombiniert. Wie CTT integriert es die Aufgaben des Nutzers (*User-Tasks*) und des Systems (*System-Tasks*) in einem Modell. Die Interaktion zwischen Nutzer und System wird durch *Interaction-Tasks* beschrieben.

System-Tasks werden auf Elemente des *FCA-Modells* abgebildet, in welchem der Anwendungsentwickler ihre Realisierung beschreibt. Interaction-Tasks hingegen werden auf Elemente im *AUI-Modell* abgebildet. Dieses generische Metamodell-Paket lässt beliebige Zielmodalitäten, Abstraktionsstufen und Interaktoren zu, um Konsistenz sowie die Portierbarkeit der Schnittstellen zu erleichtern (vgl. [KOC02]).

2.3 Toolchain

Die EMODE-Toolchain, bestehend aus Entwicklungs- und Laufzeitumgebung, implementiert die konzeptionelle Architektur der EMODE-Methodik. Sie integriert neben modellspezifischen Editoren auch Komponenten zur Modelltransformation und zur Codegenerierung. Grafische Editoren, die auf Grundlage des Graphical Editing Frameworks [GEF07] erstellt wurden, erlauben das Erstellen und Editieren von Modellen verschiedener Abstraktionsniveaus. Die transiente Verwaltung der Modelle erfolgt in einem über CORBA eingebundenen Repository. Die Transformationen von Modellen in andere Modelle (z.B. Task-to-AUI) wurden in der Transformationssprache QVT [OMG05] realisiert. Sie ermöglichen die Generierung neuer Modellelemente sowie die Aktualisierung von Modellen nach Änderungen. Bindeglied zwischen Entwicklungs- und Laufzeitumgebung ist die Modell-zu-Code-Transformation. Diese Transformation wurde mit Hilfe von Java Emitter Templates [JET04] realisiert.

Die Laufzeitumgebung unterstützt Kontextverarbeitung und enthält Komponenten zur Ein- und Ausgabe in verschiedenen Modalitäten. Dafür interpretiert eine Process-Engine (Controller) das nach Java transformierte Task-Modell. Sie steuert das Zusammenspiel von generiertem Code, Kontextdienst und Modality-Service-Komponente.

3 Beispiel

Im Rahmen von EMODE werden zwei Demonstratoren erarbeitet: ein Reiseassistent und ein Instandhaltungsszenario. Anhand eines Ausschnitts des Reiseassistentenszenarios wird die Entwicklung adaptiver, multimodaler Applikationen illustriert. Ein Autofahrer wird während der Fahrt mit Informationen über die nächstgelegene Sehenswürdigkeit versorgt. Alle Informationen werden über die akustische Modalität ausgegeben. Überschreitet der Schalldruckpegel im Fahrzeug jedoch einen Schwellwert, erfolgt die Ausgabe visuell über einen integrierten Bildschirm.

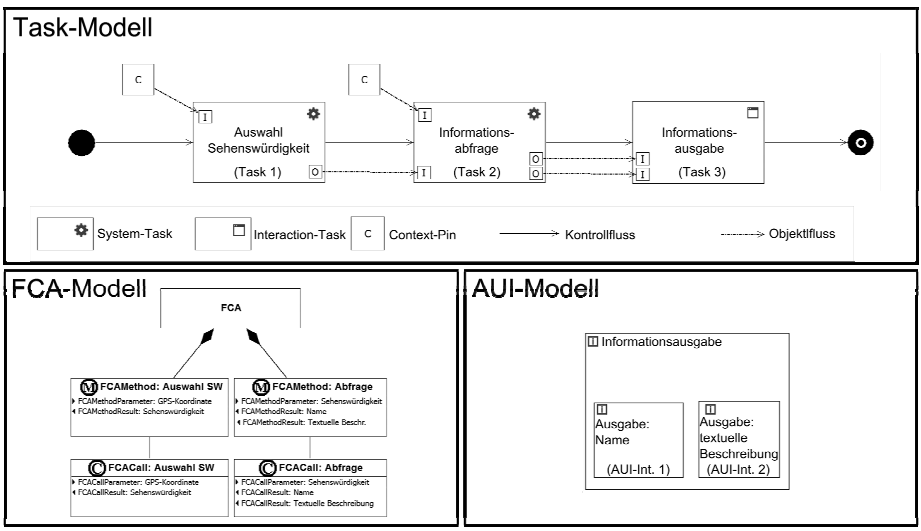

Abbildung 3: Modelle der Beispielanwendung

Der Fokus der Entwickler liegt auf drei essentiellen Modellen: Task-, FCA- und AUI-Modell. Das Task-Modell in Abbildung 3 zeigt einen die Applikation beschreibenden Kontrollfluss bestehend aus drei Tasks. Ein erster System-Task wählt in Abhängigkeit von der aktuellen GPS-Koordinate des Fahrzeugs automatisch die nächstgelegene Sehenswürdigkeit. Die GPS-Information wird über einen Context-Pin zur Verfügung gestellt. Anschließend entscheidet Task 2, basierend auf Kontextinformation (z.B. Schalldruckpegel), ob die Information zur Sehenswürdigkeit visuell oder akustisch aufbereitet an Task 3 übermittelt werden soll. Interaction-Task 3 gibt die Information entsprechend der selektierten Modalität aus.

In einem zweiten Modellierungschritt wird durch die Modell-zu-Modell-Transformation das FCA- und das AUI-Modell generiert (Abbildung 3). System-Tasks finden sich im FCA-Modell als FCA-Calls wieder, für welche die über Input- und Output-Pins angebundenen Konzepte als Ein- und Ausgabeparameter übernommen werden. Die FCA-Calls werden an FCA-Methods gebunden, deren Signaturen wiederum im Hinblick auf die Anbindung der Anwendungslogik bindend sind. Im Gegensatz zu System-Tasks werden Interaction-Tasks auf abstrakte UI-Interaktoren im AUI-Modell abgebildet. Diese abstrakten Interaktoren sind Platzhalter für die zur Laufzeit verfügbare Information.

Beispielsweise zeigt Interaktor 1 den Namen und Interaktor 2 eine textuelle Beschreibung der gewählten Sehenswürdigkeit. Die Modelle werden abschließend in Code überführt. Dieser ist im Zusammenspiel mit den existierenden Laufzeitkomponenten bereits lauffähig. Der Entwickler muss schließlich noch die Anwendungslogik implementieren.

4 Diskussion und Ausblick

In diesem Artikel wurden die EMODE-Methodik und die darauf basierende Toolchain vorgestellt und an einem Beispiel erläutert. Der Zusammenschluss, der hier vorgestellten Komponenten, unterstützt gezielt die Entwicklung multimodaler, adaptiver Anwendungen. Die EMODE-Methodik liefert die Grundlage und fokussiert eine bessere Integration der verschiedenen Teilnehmer am Entwicklungsprozess durch schrittweise Konkretisierung des zu erstellenden Systems. Dies zielt zusammen mit der vorgestellten Toolunterstützung auf eine Verbesserung der Effizienz ab. Eine iterative Entwicklung wird ebenfalls unterstützt, wodurch es jederzeit leicht möglich ist, das Feedback des Nutzers einzuholen und Veränderungen vorzunehmen.

Im weiteren Verlauf von EMODE werden verschiedene Beispielanwendungen auf Basis der EMODE-Toolchain entwickelt. Desweiteren werden die vorgestellten Komponenten und die EMODE-Methodik evaluiert, mit Fokus auf Eignung zur Effizienzverbesserung.

Das EMODE-Projekt wird durch das BMBF im Rahmen von ITEA gefördert.

Literaturverzeichnis

[CAL02] Calvary, G. Coutaz, J., Thevernin, D. Limbourg, Q., Souchon, N. Bouillon, L., Florins, M., Vanderdonckt, J.: Plasticity of User Interfaces: A Revised Reference Framework, TAMODIA, S. 127-134, 2002.

[EMO05] EMODE Consortium: Emode Website, Verfügbar unter: www.emode-projekt.de

[GEF07] GEF Project Page: www.eclipse.org/gef

[JET04] Java Emitter Templates: www.eclipse.org/articles/Article-JET/jet_tutorial1.html

[KO02] Koch, T.; Uhl, A. & Weise, D.: Model Driven Architecture, 2002.

[LIM04] Limbourg, Q.: Multi-Path Development of User Interfaces, PhD thesis, Université catholique de Louvain, 2004.

[MDA07] OMG Model Driven Architecture, Website: www.omg.org/mda, 2007.

[OMG05] MOF QVT Final Adopted Specification. www.omg.org/docs/ptc/05-11-01.pdf, 2005.

[PAT97] Paterno, F.; Mancini, C. & Meniconi, S.: ‚ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models, *in* 'INTERACT '97: Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction', Chapman & Hall, Ltd., London, UK, UK, pp. 362—369, 1997.

[ST04] Steglich, S., Mrohs, B.: A Markup Language for Generic User Interaction, The 2004 International Conference on Internet Computing (IC'04), Las Vegas, Nevada, USA, 2004.