

EMODE: Enabling Model Transformation-Based Cost Efficient Adaptive Multi-modal User Interfaces

Das EMODE-Konsortium¹

www.emode-projekt.de

Juli 2007

Kurzfassung

Das im Oktober 2005 gestartete, zweijährige Vorhaben EMODE verfolgt das Ziel, die Entwicklung adaptiver, multimodaler Anwendungen durch eine einheitliche Modellierung sowie eine integrierte Werkzeugumgebung wesentlich zu vereinfachen und damit die Entwicklungskosten zu senken, ohne die Benutzbarkeit zu beeinträchtigen. Die angestrebte Lösung basiert auf dem Ansatz der Model Driven Architecture (MDA) der Object Management Group (OMG). Mit diesem Ansatz wird ein Anwendungsmodell mittels Modelltransformationen schrittweise in eine ausführbare Anwendung überführt. Modellbasierte Ansätze sowohl aus der Softwareentwicklung als auch aus der Mensch-Maschine-Interaktion werden betrachtet und zu einer ganzheitlichen Methodik integriert. Dabei werden insbesondere die Modellierung von Kontext und Modalitäten sowie die Anwendung solcher Modelle zur Adaption von Anwendungen betrachtet. Zum jetzigen Zeitpunkt (Juli 2006) ist die Konzeptionsphase des Projekts erfolgreich abgeschlossen. Die entstandenen Spezifikationen (Metamodell, Entwicklungsmethodik, MDA-basierte Entwicklungs- und Laufzeitumgebung) werden derzeit implementiert. Die praktische Anwendbarkeit des EMODE-Konzeptes wird mittels zwei realer Anwendungsszenarien, jeweils aus der Prozess- und der Automobilindustrie, validiert.

1. Einleitung und Vorstellung des Themenkomplexes

Adaptive und multimodale Applikationen erlauben, neuartige Kommunikationsgeräte, wie z.B. PDA, mobile Telefone, tragbare Computer, zur flexiblen Interaktion mit dem Nutzer zu verwenden. Neue Modalitäten wie Sprache und Gesten versprechen einen hohen Grad der Anwenderfreundlichkeit, da sie die traditionellen Interaktionstechnologien, wie Bildschirm, Tastatur und Maus, erweitern bzw. so gar ersetzen, die typischerweise volle Konzentration und beide Hände des Benutzers voraussetzen.

Derzeit gestaltet sich die Entwicklung adaptiver, multimodaler Anwendungen aufgrund fehlender standardisierter und effizienter Methoden sowie integrierter Werkzeuge komplex und kostenintensiv. Als Folge ist ein erheblicher Aufwand zur Wahrung der Konsistenz zwischen den Modellwelten notwendig. Die Nutzung der heute verfügbaren Methoden und Werkzeuge

¹ BASF AG (M. Schwibach), CyberConsult GmbH (J. Landvogt, H. Hoffmann), DaimlerChrysler AG (W. Enkelmann, S. Thakar), IKV++ AG (O. Kath, J. Höbner), TU Darmstadt (A. Behring, A. Petter, J. Steinmetz), TU Dresden (T. Hamann, G. Hübsch, R. Neumerkel), SAP AG (K. Fetzer, M. Winkler, H.H. Do); Koordinator-kontakt: Karin Fetzer, SAP AG, karin.fetzer@sap.com

führt meist zu plattformspezifischen Lösungen, die insbesondere eine Wiederverwendung verhindern und nachträgliche Änderungen bzw. Erweiterungen erschweren. Eine fehlende integrierte Werkzeugumgebung erhöht die Entwicklungszeit und damit die Kosten zusätzlich. Das Vorhaben EMODE [1] hat die Vereinfachung der Entwicklung adaptiver, multimodaler Anwendungen und damit verbunden die Senkung der Entwicklungskosten zum Ziel. EMODE baut auf der Model Driven Architecture (MDA) [17] der Object Management Group (OMG) auf. Dabei wird ein Modell als Abstraktion verstanden, das durch eine Anreicherung mit Informationen über die Ausführungsplattform in Form von Transformationen schrittweise verfeinert und dadurch semi-automatisch in ein lauffähiges Softwaresystem überführt werden kann. Die große Herausforderung besteht darin, modellbasierte Ansätze aus der Softwareentwicklung und aus der Mensch-Maschine-Interaktion (MMI) zu integrieren und um Methoden der Adaption und Multimodalität zu erweitern. Hauptziele von EMODE sind:

- Entwicklung eines Metamodells zur einheitlichen Modellierung unterschiedlicher Aspekte (fachliche Logik, Nutzerschnittstelle) einer adaptiven, multimodalen Anwendung.
- Entwicklung einer modellbasierten Software-Entwicklungsmethodik für adaptive, multimodale Applikationen. Dabei erfolgt eine Integration der Benutzungsschnittstellenentwicklung und eine Einbeziehung von Kontextinformationen zur Adaption.
- Implementierung der Methodik in Form einer Entwicklungs- und Laufzeitumgebung. Während die Entwicklungsumgebung die Werkzeuge zur Modellierung und Entwicklung einer Anwendung bereitstellt, bietet die Laufzeitumgebung die notwendige Infrastruktur zur Ausführung einer Anwendung.
- Zur Validierung und Verbreitung der Projektergebnisse werden zwei Anwendungen, jeweils aus der Prozess- und der Automobilindustrie, entwickelt.

Für die Realisierung der EMODE-Vision kann auf eine Vielzahl von State-of-the-Art-Technologien zurückgegriffen werden. Für die Modellierung von Anwendungen hat sich UML [14] weitgehend durchgesetzt und wird im Rahmen der MDA [10] verwendet. Zur Modellierung von Interaktionen existieren verschiedene Ansätze aus dem MMI-Bereich, z.B. ConcurTaskTrees (CTT) [9], die jedoch meist wenig verbreitet sind. Für die Beschreibung multimodaler Interaktion existieren unter anderem diverse XML-basierte Sprachen wie X+V [17] bzw. Extensible Multimodal Annotation Markup Language (EMMA) [16]. Für die Modellierung und Beschreibung von Kontext gibt es verschiedene Standards auf unterschiedlichen Abstraktionsstufen, z.B. CC/PP [15] für Modellierung der Fähigkeiten von Endgeräten.

EMODE ist thematisch verwandt mit einigen Forschungsprojekten, u.a. SMARTKOM [5], EMBASSI [4] und EAST-EEA [3] (abgeschlossen), sowie DYNAMITE [2], SMARTWEB [6], USEKIT [8] und TOPPRAX [7] (aktuell). Wesentliche Unterschiede bestehen jedoch in den Schwerpunkten der Projekte. EMODE verfolgt die Konzeption, Umsetzung und Erprobung einer ganzheitlichen, allgemeingültigen Software-Entwicklungsmethodik für multimodale, adaptive Anwendungen, während die anderen Projekte entweder auf die Realisierung spezifischer multimodaler Systeme oder auf einzelne technische Aspekte fokussieren.

Dieser Beitrag gibt einen Überblick über den aktuellen Stand des Projekts (nach neuen Monaten von der zweijährigen Gesamtlaufzeit). Abschnitt 2 beschreibt den Fortschritt des Projekts in Bezug auf die einzelnen genannten Ziele. Abschnitt 3 berichtet von Erfahrungen der Projektpartner in der bisherigen Projektlaufzeit sowie im Umgang mit relevanten Technologien. Abschnitt 4 gibt einen Ausblick.

2. Projektstatus

In den ersten 9 Monaten der Projektlaufzeit wurden die folgenden wesentlichen Ergebnisse erreicht:

- Das integrierte Metamodell zur einheitlichen Modellierung der fachlichen Logik und multimodalen Nutzerschnittstelle von adaptiven, multimodalen Applikationen wurde entwickelt.
- Die modellbasierte Software-Entwicklungsmethodik für adaptive, multimodale Anwendungen wurde unter Berücksichtigung traditioneller MMI- sowie MDA-basierter Ansätze konzipiert.
- Nach einer umfassenden Anforderungsanalyse wurden detaillierte Spezifikationen der Entwicklungs- und Laufzeitumgebung erstellt. Die Spezifikationen liegen nun der Implementierung der entsprechenden Umgebung zugrunde.
- Die Anwendungen wurden detailliert mit notwendigen Hardware- und Software-Komponenten spezifiziert. Die Spezifikationen dienen als Quelle für Anforderungen für technologische Entwicklungen des Projektes.

Diese Ergebnisse werden in den folgenden Unterabschnitten vorgestellt und diskutiert.

2.1 Das EMODE-Metamodell

Durch die heute übliche getrennte Modellierung von fachlicher Logik und Benutzungsschnittstelle treten verschiedenste Probleme auf. So können etwa vorhandene Vorarbeiten aus der Modellierung der fachlichen Logik nicht verwendet werden, da die Benutzungsschnittstelle i.d.R. weitestgehend unabhängig davon modelliert wird. Die Konsistenz von Benutzungsschnittstelle und fachlicher Logik muss meist manuell überprüft werden. Daher verfolgt EMODE die Entwicklung eines integrierten Metamodells zur einheitlichen Modellierung unterschiedlicher Aspekte einer Anwendung (fachlicher Problembereich, Benutzungsschnittstelle, Kontext).

In EMODE wurde bereits ein solches Metamodell entwickelt, das während der Projektlaufzeit weiter angepasst wird. Das Metamodell ist modular aufgebaut, um es einfach erweitern und anwenden zu können. Es besteht aus mehreren sog. *Packages*, die einzelne Aspekte einer adaptiven und multimodalen Anwendung adressieren und Mittel zur Modellierung des Aspektes bereitstellen. Zur Erstellung des Modells der Anwendung, werden die in den Packages definierten Elemente herangezogen und in einem Modell eingesetzt.

- **Goal.** Mit Hilfe der Komponenten des *Goal-Package* wird das Goal-Modell (durch einen Entwickler) erstellt. Dieses umfasst die Ziele, die bei der Anforderungsanalyse für die zu entwickelnde Anwendung identifiziert wurden. Ein Beispiel für ein solches Ziel wäre z.B. die Mobile Erfassung der Anlagendaten bei der Instandhaltung, welche u.a. in einem der zwei Anwendungsszenarien von EMODE verfolgt wird.
- **Task.** Mit den Komponenten aus dem *Task-Package* definiert der Entwickler, welche Aufgaben (Tasks) im Zusammenhang der Anwendung durchgeführt werden müssen, um die im Goal-Modell spezifizierten Ziele zu erreichen. Diese Aufgaben sind in System, Interaction und User aufgeteilt, um nach den verschiedenen ausführenden Entitäten zu unterscheiden.
- **DomainConcept.** Das *DomainConcept-Package* enthält die Konzepte (z.B. Entitäten), die bei der Modellierung relevant sein könnten und zur Darstellung von Daten aus der Sicht der Applikation dienen. Hierbei können insbesondere die Interaktoren mit Konzepten verbunden werden und können so ihre Informationen mit Komponenten aus anderen Packages teilen.
- **Abstract User Interface.** Das *AUI-Package* bietet die Elemente für Nutzerschnittstellen

zugrunde liegende Komponententechnologie an. Hier können Interaktoren und deren Beziehungen dargestellt werden. Kernidee des Packages ist die Darstellung von kleineren agierenden Komponenten, den Interaktoren, in einer größeren technischen Einheit, die die Darstellung der einzelnen Komponenten kapselt und damit die eher einfach agierenden Komponenten zu einem kompletten multimodalen Dialog zusammenführt.

- **Context.** Das *Context-Package* dient zur Abstraktion und Integration von Sensorinformationen in Anwendungen. Hierbei werden Push und Pull Techniken unterschieden. Während Push eventbasiert Informationen liefert, kann Pull von der Applikation gesteuert abgefragt werden.
- **Common.** Das *Common-Package* enthält eine Vielzahl von Metamodellelementen, die für die anderen Packages notwendig sind. Hierbei wird auch auf die Möglichkeit eingegangen, Design-Pattern zu spezifizieren und darzustellen. In Zukunft erwarten wir neue Design-Pattern für multimodale adaptive Anwendungen, die mit Hilfe des EMODE Metamodells dargestellt und identifiziert werden könnten.

Das Metamodell basiert auf dem weit verbreiteten Standard für Metamodellierung MOF [13], der von der OMG (Object Management Group) spezifiziert wurde. Darauf aufbauend definierte OMG ebenfalls auch den Modellierungsstandard UML. UML dient EMODE als eine Art Bibliothek und als Startpunkt für Erweiterungen. Einige speziell definierte Elemente des Metamodells lassen sich in Kombination mit aus UML bekannten Elementen verwenden, wodurch eine Integration in bestehende Modelle erleichtert wird.

2.2 Die EMODE-Entwicklungsmethodik

Für den Entwicklungsprozess adaptiver, multimodaler Anwendungen wurde eine Methodik herausgearbeitet. Dabei wurde eine umfangreiche State-of-the-Art-Analyse der existierenden Software-Entwicklungsmethoden (das Wasserfall-Modell, V-Modell, Extreme Programming, Unified Process), der Prozessmodelle aus dem Gebiet der MMI (Usability Engineering) sowie der MDA-Methoden (Executable UML) durchgeführt. Darüber hinaus wurden die Entwicklungsprozesse der Partner wie CEA, SAP, IKV++, Intuilab und Robotiker analysiert, die umfangreiches Know-how in den Bereichen der multimodalen/mobilen Anwendungen bzw. der modellbasierten Software-Entwicklung vorweisen. Die Vor- und Nachteile der Ansätze wurden bei der Entwicklung der EMODE-Methodik berücksichtigt. Ähnlich wie das Metamodell wird die Methodik kontinuierlich während der Projektlaufzeit mit neuen Ergebnissen und Erkenntnissen verbessert.

Wie Abbildung 1 zeigt, basiert die EMODE-Methodik auf der Iteration traditioneller Phasen des Wasserfall-Vorgehensmodells, nämlich *Anforderungsanalyse*, *High-Level Design*, *Detailed Design*, und *Implementierung*. Gemäß dem MDA-Ansatz definiert sie, wie in Bezug auf das Metamodell valide Anwendungsmodelle erstellt und schrittweise verfeinert werden können und wie nachträgliche Änderungen auf höherer Ebene in konkretere Modelle propagiert werden. Ferner integriert sie Elemente aus dem MMI-Bereich, nämlich die Modellierung der abstrakten und die Generierung der finalen Nutzerschnittstelle. Im Folgenden werden die Phasen der EMODE-Methodik kurz vorgestellt:

Anforderungsanalyse. In dieser Phase werden Anforderungen bezüglich der Benutzerschnittstelle und der Anwendungslogik identifiziert. Insbesondere werden funktionale und nicht-funktionale Anforderungen in Zusammenarbeit mit zukünftigen Anwendern gesammelt und analysiert, um eventuelle Abhängigkeiten und Konflikte zwischen den Anforderungen zu identifizieren. Die Anforderungen führen zur Bestimmung konkreter Ziele für die Anwendung, die dann im *Goal-Modell* dargestellt werden.

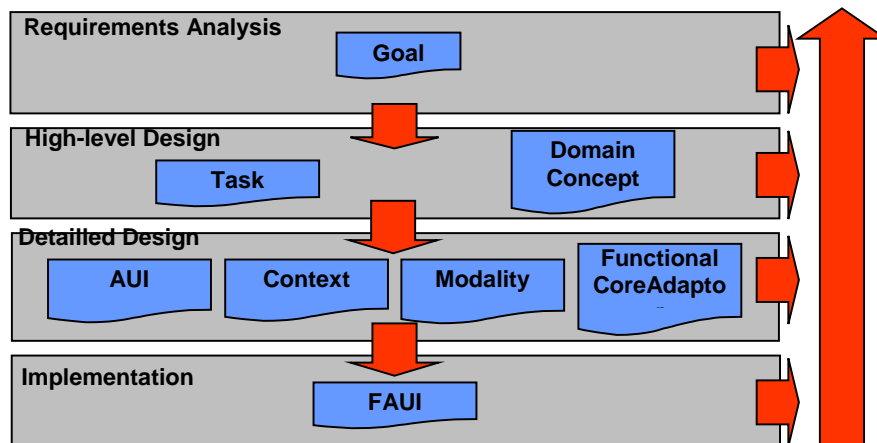


Abbildung 1: Die EMODE-Methodik

High-level Design. Auf Basis der im *Goal-Modell* spezifizierten Ziele werden in dieser Phase zunächst die Aufgaben (Tasks) identifiziert, die die Anwendung zur Erreichung der entsprechenden Ziele leisten soll. Bei der Spezifikation der Aufgaben werden die Interaktionen des Benutzers mit der Anwendung auf eine plattform- und geräteunabhängige Art und Weise modelliert. Die Aufgaben bilden nun das *Task-Modell* der Anwendung. Ferner werden die Konzepte (z.B. Entitäten) modelliert, die zur Darstellung von Daten an der Nutzerschnittstelle bzw. zur Verarbeitung der Daten durch die Applikation dienen. Diese ebenfalls plattform- und geräteunabhängigen Konzepte werden im *DomainConcept-Modell* der Anwendung abgelegt.

Detailed Design. Diese Phase beschäftigt sich weiterhin mit der plattform- und geräteunabhängigen Modellierung der zu entwickelnden Anwendung. Anhand der Task- und Domain-Concept-Modelle werden vier plattform- und geräteunabhängige Modelle (PIM – Platform-Independent Models) abgeleitet. Das *AUI-Modell* beinhaltet die abstrakte, von den physischen Interaktoren der Endgeräte unabhängige Benutzungsschnittstelle. Das *Context-Modell* beschreibt wie die Applikation Kontext-Informationen gewinnen kann. Das *Modality-Modell* spezifiziert, welchen Anforderungen die Modalitäten genügen müssen um bestimmte Interaktionen durchführen zu können. Schließlich beschreibt das *FunctionalCoreAdapter-Modell* die zugrunde liegende Funktionalität (d.h. fachliche Logik) der Anwendung.

Implementation. Diese Phase beinhaltet den Übergang von plattform-/geräteunabhängigen Modellen zu plattform-/geräteabhängigen Modellen (PSM – Platform-specific Models) und schließlich zum ausführbaren Code der Anwendung. Mittels der in EMODE entwickelten Modelltransformationen werden die plattform- und geräteunabhängigen Modelle schrittweise mit Plattform- und Geräteinformationen angereichert. Gemäß dem MDA-Ansatz erfolgt diese Phase iterative, wodurch mehrere Iterationen notwendig sein und Modelle mit unterschiedlichem Abstraktionsgrad bzgl. Plattform und Geräteabhängigkeit entstehen können. In jeder Iteration werden gemeinsame/ähnliche Eigenschaften einer Teilmenge von Plattformen bzw. Geräte berücksichtigt, so dass die daraus resultierenden Modelle für speziellere Anpassung in der nächsten Iteration weiter verwendet werden können. Aus dem AUI-Modell entsteht z.B. das sog. *Final Abstract User Interface-Modell* (FAUI), das die zu unterstützenden Interaktionen auf die geräte-/plattform-spezifischer Interaktoren, wie z.B. Sprachenein- und -ausgabe eines mobilen Gerätes, abbildet.

2.3 Die EMODE-Entwicklungsumgebung

Eine zentrale Aufgabe von EMODE ist, eine Werkzeugumgebung für die Entwicklung von adaptiven multimodalen Anwendungen bereitzustellen. Entsprechend MDA wird dabei eine

Verfeinerung der anwendungsspezifischen Modelle durch Modelltransformationen unterstützt, die nahezu lauffähigen Code erzeugen sollen. Die anwendungsspezifischen Modelle sollen einheitlich mit dem beschriebenen Metamodell spezifiziert werden. In einer Anforderungsanalyse wurden die folgenden wesentlichen Funktionalitäten der Entwicklungsumgebung identifiziert:

- die Erzeugung, Bearbeitung, Visualisierung und Validierung von Modellen;
- die Speicherung, Versionsverwaltung und der Austausch von Modellen;
- die Definition und die Ausführung von Transformationen auf den Modellen.

Auf Basis dieser Funktionalitäten wurde eine Architektur für die Entwicklungsumgebung Entworfen. Abbildung 2 gibt einen Überblick über die Komponenten der EMODE-Entwicklungsumgebung. Die Komponenten und die Schnittstellen zwischen ihnen wurden bereits detailliert spezifiziert. Als nächster Schritt im Projekt sind die Modell-Editoren zu implementieren. Besondere Aufmerksamkeit gilt der Integration der Editoren und der Transformation.-Engine.

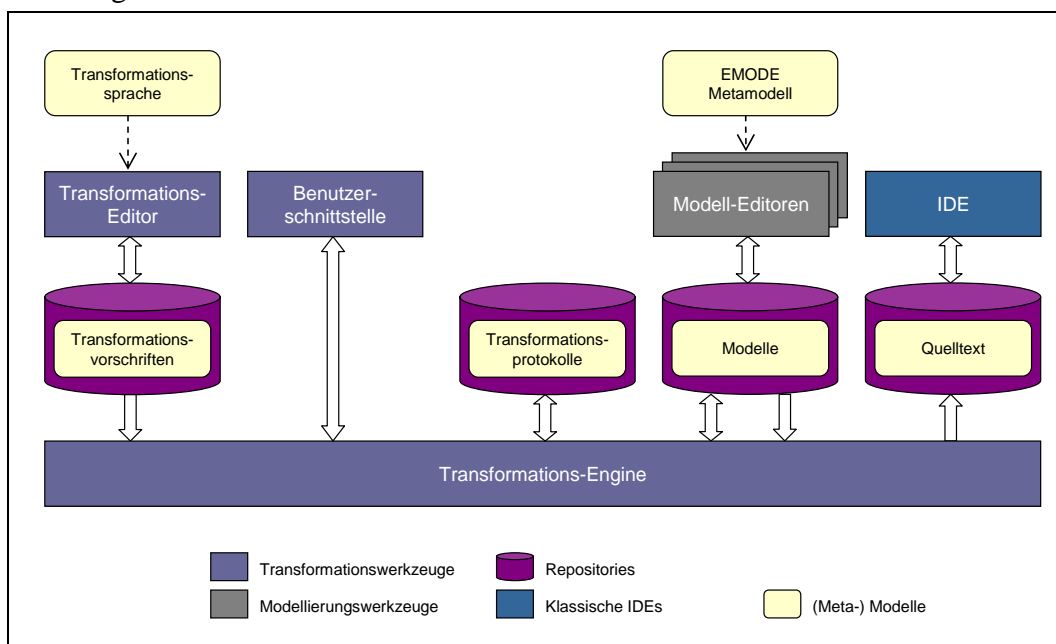


Abbildung 2: EMODE Entwicklungsumgebung

Die Komponenten der EMODE-Entwicklungsumgebung lassen sich in vier Klassen einteilen, auf die im Folgenden kurz eingegangen wird.

Modellierungswerkzeuge. Die Modell-Editoren ermöglichen das Anfertigen von Modellen eines Softwaresystems unter Verwendung der im Metamodell definierten Konzepte. Dabei wird das zu entwickelnde System aus verschiedenen Blickwinkeln sowie auf unterschiedlichen Abstraktionsebenen modelliert. Je nach Blickwinkel und Abstraktionsebene definiert die Methodik unterschiedliche Arten von Modellen, welche jeweils spezielle Konzepte des Metamodells verwenden. Demzufolge wird es in EMODE auch verschiedene Modell-Editoren geben. Diese werden jedoch eine gemeinsame Plattform (Eclipse) verwenden.

Transformationswerkzeuge. Dem MDA-Ansatz entsprechend erfolgt der Wechsel zwischen verschiedenen Abstraktionsebenen, bis zu ausführbarem Code, mittels automatischer Modell-Transformationen. Unter Verwendung der Transformationssprache QVT [12] werden entsprechende Transformationsvorschriften definiert. Darüber hinaus bietet ein Editor, die Möglichkeit, eigene Transformationsvorschriften zu definieren. Die Anwendung von Transforma-

tionsvorschriften auf Modelle erfolgt mittels einer Transformations-Engine. Diese ist in der Lage, für die in den Transformationsvorschriften definierten Regeln entsprechende Algorithmen auszuführen, die aus einem Eingangsmodell ein entsprechendes Ausgangsmodell erzeugen.

Klassische IDEs. Durch die wiederholte Anwendung von Transformationen und manueller Bearbeitung entstehen nach und nach zahlreiche Modelle mit immer niedrigerem Abstraktionsgrad. Im Idealfall lässt sich aus den Modellen direkt lauffähiger Maschinencode erzeugen. In der Praxis ist jedoch oft ein zusätzlicher Schritt notwendig. Hierbei wird aus den Modellen zunächst Programmcode generiert, der vom Entwickler noch manuell vervollständigt und angepasst werden kann. Für diesen Fall ist in EMODE die Verwendung bereits existierende Entwicklungswerkzeuge wie beispielsweise Eclipse JDT², CDT³ usw. vorgesehen.

Managementwerkzeuge. Bei der Entwicklung von Anwendungen unter Verwendung der hier beschriebenen Werkzeuge entstehen zahlreiche Artefakte in Form von Modellen, Transformationsvorschriften, Transformationsprotokollen und Quelltext. Diese werden in speziellen Repositories gespeichert, welche neben der Versionsverwaltung auch den gemeinsamen Zugriff auf die Artefakte erlauben.

2.4 Die EMODE-Laufzeitumgebung

Um die Ausführung von adaptiven multimodalen Anwendungen zu ermöglichen, welche mit der zuvor beschriebenen Entwicklungsumgebung modelliert und spezifiziert wurden, ist eine entsprechende Ausführungsumgebung zu entwickeln und zu realisieren. Ausgehend vom aktuell definierten Metamodell wurden Anforderungen an eine solche Umgebung definiert. Auf Basis dieser Anforderungen konnten wesentliche Bestandteile der Ausführungsumgebung identifiziert und abstrakt beschrieben werden.

Adaptive, multimodale Anwendungen zeigen ihre Stärken vorwiegend in Szenarien mit häufig wechselnden Umgebungsbedingungen und den damit verbundenen eingeschränkten Interaktionsmöglichkeiten des Nutzers. Daraus erwachsen zwei Grundforderungen an Anwendungen und zugrunde liegende Ausführungsumgebungen: (1) Erfassung der Umgebung, in welcher die Nutzung der Anwendung und des ausführenden Gerätes gerade erfolgt; (2) Unterstützung multimodaler Interaktion des Nutzers mit der Anwendung, d.h. Verwendung mehrerer Modalitäten (z.B. Gesten, Sprache, Haptik), um Ein- und Ausgabe von Informationen zu verbessern oder erst zu ermöglichen, und (3) dynamische Anpassung der Anwendung bzgl. Funktionalität und unterstützter Modalitäten in Abhängigkeit von der aktuellen Umgebung (Context Awareness).

Aus diesen Überlegungen leitet sich die in Abbildung 3 gezeigte Architektur der EMODE Laufzeitumgebung ab. Die *Multimodality-Services-Component* (MSC) stellt Anwendungen die Dienste bereit, die für eine multimodale Interaktion mit dem Nutzer notwendig sind. Dazu gehören Eingabemodalitäten (Input Modality Services) wie Sprache oder Gesten und Ausgabemodalitäten (Output Modality Services) wie beispielsweise taktiles Feedback. Die MSC ist in Bezug auf Art und Anzahl der unterstützten Modalitäten erweiterbar. Aufgrund der kontextabhängigen Verfügbarkeit von Modalitäten muss ihre Auswahl zur Laufzeit erfolgen. Die Modality Fission Komponente ist für die Selektion der letztendlich verwendeten Ausgabemodalitäten zuständig. In einem weiteren Schritt zerlegt die Modality Fission Komponente darzustellende Information in modalitätsspezifische Bestandteile, welche zur Darstellung an die entsprechenden Output Modality Services übergeben werden. Die Modality Fusion Kompo-

² Java Development Tools

³ C/C++ Development Tooling

nente korreliert zusammengehörige, monomodale Eingaben – beispielsweise kann eine Geste zur Verdeutlichung ihrer Bedeutung von einem Sprachkommando begleitet werden.

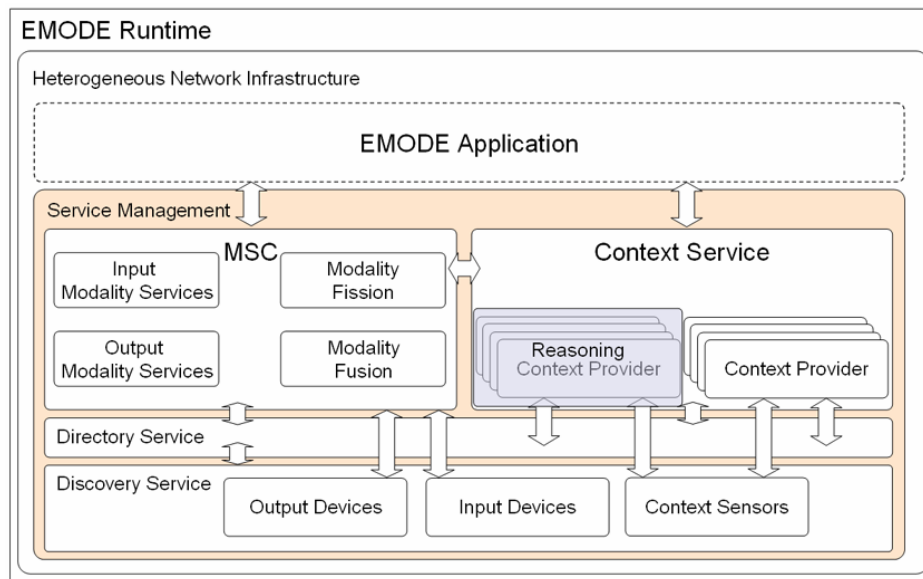


Abbildung 3: Konzeptionelle Architektur der EMODE Laufzeitumgebung

Die *Context-Service-Komponente* als zweiter Kernbestandteil der Architektur ist für die Bereitstellung von Kontextinformation zuständig. Auf Basis der von ihr gelieferten Daten können Anwendungen bzw. die MSC die vorliegende Situation bestimmen und ihr Verhalten entsprechend anpassen. Die Gewinnung von Kontext erfolgt dabei entweder über eine direkte Messung per Sensor (Context Sensors) oder durch Reasoning, d.h. der Ableitung höherwertiger Kontextinformation auf Basis von Sensordaten. Jegliche Art von Datenquelle, die aus Sicht einer Anwendungen Kontextinformation liefert, wird durch einen Kontextanbieter (Context Provider) gekapselt. Die Context Service Komponente verwaltet diese Kontextanbieter und stellt den zentralen Zugangspunkt zu Kontextinformation innerhalb der Architektur dar.

Dem Wissen über die Verfügbarkeit von Ein- und Ausgabegeräten sowie von Kontextsensoren und deren dynamischer Auswahl kommt in mobilen Szenarien eine besondere Bedeutung zu. Aufgrund dessen existiert zwischen diesen Hardware-Komponenten und der EMODE Architektur keine statische Bindung. Vielmehr werden Mechanismen zur automatischen Ermittlung verfügbarer Hardware unterstützt. Ein entsprechender Dienst (Discovery Service) ist für das Auffinden und Einbinden von I/O-Geräten und Kontextsensoren zuständig. Gefundene Geräte und Sensoren werden im Verzeichnisdienst (Directory Service) registriert. MSC und Context Service verwenden den Verzeichnisdienst ihrerseits, um nach verfügbaren Geräten zu suchen.

2.5 Anwendungsszenarien

Zur Validierung der EMODE-Konzepte werden zwei Anwendungen spezifiziert und entwickelt, ein adaptive Reiseassistent im Fahrzeug (von DaimlerChrysler AG) und eine mobilen Anwendung in der Instandhaltung der Chemieproduktionsanlagen (von BASF AG) durchgeführt werden.

Mobile Instandhaltung (BASF AG)

Das zu betrachtende Szenario beschreibt, wie die Instandhaltung von Anlagen in der chemischen Industrie durch den Einsatz von mobilen Technologien unterstützt werden kann. Ziel ist es, die bestehende Systeminfrastruktur und die Abläufe so miteinander zu verknüpfen, dass

ineffiziente Arbeitsschritte (z.B. nachträgliches Einpflegen von manuell erfassten Daten) vermieden werden. Dabei sind die Hauptziele die Effizienzsteigerung durch mehr Flexibilität, die Minimierung der Stillstandzeiten von Anlagen, die schnelle und zielgerichtete Einbindung von internen und externen Experten und die Automatisierung der Arbeitsabläufe bzw. die Unterstützung durch eine geeignete Systeminfrastruktur sowie ein effizientes Controlling der durchgeführten Maßnahmen.

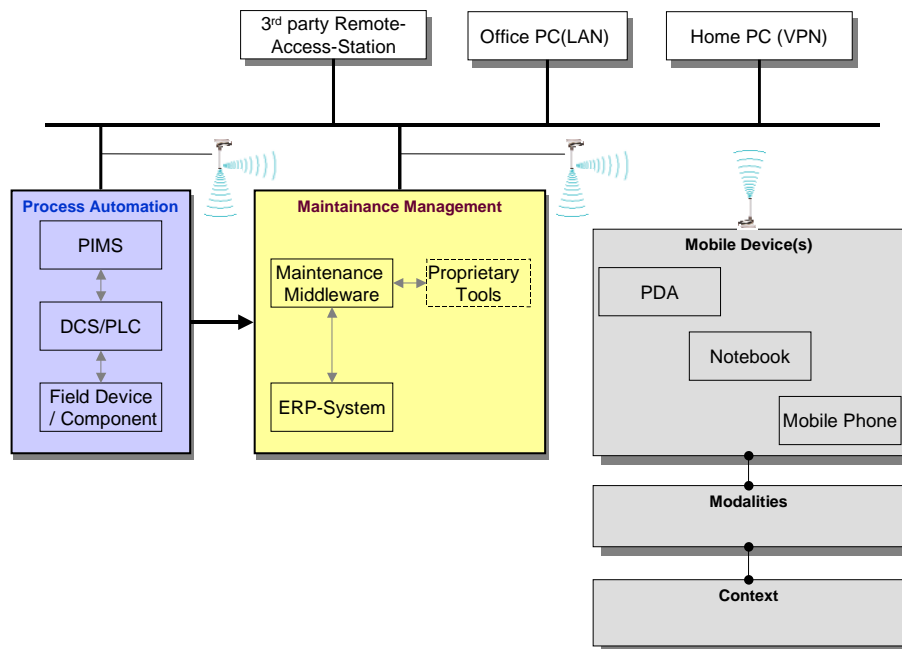


Abbildung 4: Systemkomponenten der Mobilen Instandhaltung

Der Einsatz in der Prozessindustrie ist durch Arbeiten in einem schwierigen Arbeitsumfeld gekennzeichnet. Ein Teil der Anlagen sind Freiluftanlagen. Die Arbeiten dort müssen bei jeglichen Witterungsverhältnissen durchgeführt werden. Für manche Arbeiten ist Schutzkleidung vorgeschrieben, da die Arbeitsstoffe giftig oder ätzend sein können. Dies führt zu Einschränkungen bei der Auswahl des mobilen Gerätes, aber auch bei den Möglichkeiten der Handhabung zum Abruf und zur Eingabe von Informationen. Das gleiche gilt für Arbeitsbereiche, in denen mit explosiver Atmosphäre gerechnet werden muss oder für laute Arbeitszonen, in denen das Tragen eines Gehörschutzes Pflicht ist. Je nach Einsatzziel muss damit für die gleiche Aufgabe ein für diese Anforderungen zugelassenes Mobilsystem eingesetzt werden.

Die mobile Anwendung muss verschiedene Zielgruppen mit verschiedenen mobilen Endgeräten unterstützen. Diese Zielgruppen sind: Handwerker, Spezialisten, Meister und Manager. Die von den Zielgruppen heute verwendeten mobilen Endgeräte lassen sich in folgende Klassen einteilen: Handy, PDA, Spezialsystem und Notebook / Tablet-PC. Entsprechend müssen die Anwendungen hinsichtlich ihrer Nutzerschnittstellen diese verschiedenen Kombinationen unterstützen, ohne dass aufwändige Neuprogrammierung der Anwendungen erforderlich ist. Abbildung 4 gibt einen Überblick über die Systemkomponenten des Szenarios und beschreibt die Schnittstellenanforderungen.

Adaptiver Reiseassistent im Fahrzeug (DaimlerChrysler AG)

Ein adaptiver Reiseassistent ist eine Softwarekomponente, die den Passagieren eines Fahrzeugs und insbesondere dem Fahrer vor, während und eventuell auch nach einer Fahrt reisebezogene und situations- sowie nutzerangepasste Assistenzdienste zur Verfügung stellt. Dazu

verwendet der Reiseassistent sowohl Informationen, die lokal im Fahrzeug vorliegen, als auch Informationen, die durch externe Systeme bereitgestellt werden. Auswahl, Filterung und Präsentation der Informationen erfolgen abhängig von der aktuellen Situation (beschrieben durch Kontextinformationen) und von den Nutzerpräferenzen. Für die Situationsanpassung werden auch unterschiedliche Geräte und Bedienmöglichkeiten betrachtet.

Für die Implementierung des Reiseassistenten wird vom folgenden Szenario ausgegangen. Ein Benutzer möchte im Rahmen einer Reise eine Besichtigungstour planen. Dazu greift er zunächst über einen ortsfesten Rechner (im Büro oder daheim) auf ein Portal eines Dienstansbieters zu, über das verschiedene Informationsassistenten angeboten werden, deren Eigenschaften multimodal erlebbar gemacht werden sollen. Aus diesem Angebot wählt der Nutzer den Reiseassistenten aus. Dieser bietet verschiedene vorbereitete Touren an, über die sich ein Nutzer näher informieren kann. Nun entscheidet er sich für eine Tour und kann diese im Tourplanungsmodus unter Beachtung der Planungssituation (z. B. Art der Reise, Zwischenziele) an seine individuellen Vorlieben und Randbedingungen anpassen. Das Ergebnis dieser Benutzerinteraktion ist eine geplante Tour.

Bei Antritt der Fahrt wird der Reiseassistent im Fahrzeug aktiviert, um während der Reise entlang der Tour personalisierte und situationsangepasste Informationen zu geben. Der Situationskontext wird dabei aufgrund verschiedener Sensor- und Datenquellen erkannt. Dazu gehören z. B. im Fahrzeug über den CAN-Bus verfügbare Daten wie Geschwindigkeit, Tankfüllstand, Außentemperatur, die aktuelle geografische Position sowie während der Fahrt erhaltenen Meldungen.

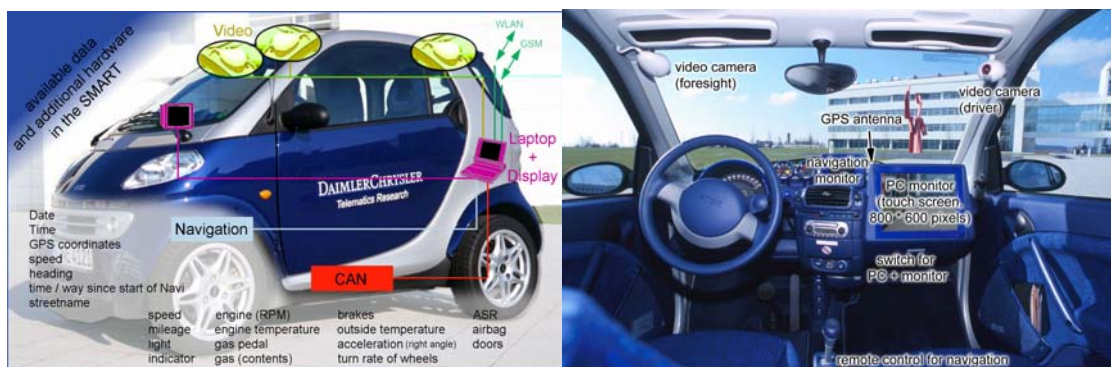


Abbildung 5: Komponenten des Fahrzeug-Reiseassistenten

Die Informationsdarstellung soll sowohl auf tragbaren Geräten wie z. B. einem Smart Phone oder PDA als auch über verschiedene Medien und Geräte im Kraftfahrzeug, insbesondere auch einer Kombination von Geräten, erfolgen können. Abbildung 5 zeigt ein Experimentierfahrzeug mit entsprechenden integrierten Geräten zur Datenerfassung und zur Interaktion mit einem Nutzer.

3. Erfahrungen, Bewertungen

EMODE ist eine ITEA⁴-Initiative. Neben dem deutschen Konsortium existieren Konsortien in anderen europäischen Ländern, nämlich Frankreich, Spanien, und den Niederlanden, die ebenfalls die EMODE-Ziele im Rahmen der jeweiligen nationalen Finanzierungsmöglichkeiten verfolgen. Die unterschiedlichen Rahmenbedingungen der Länder und der Konsortien führten zu Maßnahmen, die Voraussetzung für eine effektive Kooperation sind.

⁴ ITEA (Information Technology for European Advancement), www.itea.org

Die unterschiedlichen Größen der Konsortien führen zur Notwendigkeit einer verstärkten Zusammenarbeit. Die deutschen und französischen Konsortien sind mit jeweils 9 und 10 Partnern stark vertreten und dadurch in der Lage, auf nationaler Ebene die EMODE-Ziele zu verfolgen. Dagegen sind die Niederlande und Spanien mit jeweils 1 und 2 Partnern stark auf Kooperation mit Partnern aus anderen Ländern angewiesen. Regelmäßige Treffen auf europäischer Ebene werden für EMODE organisiert und von meisten Partnern besucht. Erst dadurch sind eine Harmonisierung der fachlichen Begriffe und Konsolidierung der Ergebnisse möglich. Der Stand bzgl. technologischen Know-how und Projektfortschritt werden regelmäßig zwischen den Konsortien abgeglichen, so dass Möglichkeiten zu Synergien, Austausch, und Ergänzung identifiziert und ausgenutzt werden können. Aufgrund der breiten wissenschaftlichen Basis der Kooperation kann die Akzeptanz der Forschungsergebnisse des gesamten europäischen Konsortiums erhöht werden.

Aus fachlicher Sicht konnten die Partner bereits umfangreiche Erfahrungen mit unterschiedlichen UI- und MDA-Technologien sammeln, insbesondere der Metamodellierung, Modellmanagement und -transformation sowie Erfahrungen mit der Integration dieser Technologien. Einer der Kernpunkte, welcher zur Metamodellierung zu entscheiden ist, betrifft die Abstraktion des Metamodells. Grob lässt sich zwischen direkt anwendungsbezogen gegenüber generalistisch unterscheiden. Es gilt hier die Vor- und Nachteile der beiden Ansätze abzuwägen. Insbesondere auch im Hinblick auf die Laufzeit- und Entwicklungsumgebung. Um auf der einen Seite dem Entwickler genügend Möglichkeiten und Ausdruckstärke zu bieten und gleichzeitig auf der anderen Seite die Entwicklungs- und Laufzeitumgebung in realisierbaren Grenzen zu halten, wurde bei EMODE ein Ansatz in der Mitte gewählt. Hierzu wurde intensiv auch im Hinblick auf die Implementierbarkeit der Konzepte diskutiert.

Das EMODE Metamodell ist als ein sehr allgemeines Metamodell für die Erstellung multimodaler, adaptiver Anwendungen geplant worden. Da EMODE die sich stellenden Probleme von multimodalen adaptiven Anwendungen auf einer allgemein gültigen Basis angehen will, bietet es sich für ein Unternehmen an, die in EMODE definierten Packages zu spezialisieren und weitere mehr auf die speziellen Problemstellungen des Unternehmens ausgerichtete Packages hinzuzufügen, wenn es mehrere ähnliche auf den EMODE Erkenntnissen basierende Applikationen erstellen will. Dadurch wird der Entwicklungsaufwand von multimodalen adaptiven Anwendungen weiter reduziert und gleichzeitig die Fehleranfälligkeit entsprechend definierter Anwendungen reduziert. Das EMODE Metamodell basiert auf MOF und kann daher leicht entsprechend spezialisiert werden. Erweiterungen sind leicht möglich, da Schnittstellen zu UML vorgesehen wurden.

4. Ausblick

EMODE verfolgt das (für eine relativ kurze Laufzeit von 2 Jahren ehrgeizige) Ziel, die Entwicklung adaptiver, multimodaler Anwendungen durch eine einheitliche Modellierung sowie eine integrierte Werkzeugumgebung wesentlich zu vereinfachen und damit die Entwicklungskosten zu senken. Zum jetzigen Zeitpunkt (Juli 2006) ist die Konzeptionsphase des Projekts erfolgreich abgeschlossen. Die entstandenen Spezifikationen werden derzeit implementiert. Die Entwicklungsumgebung wird dann anhand von zwei Anwendungen validiert.

Referenzen

1. EMODE. www.emode-projekt.de
2. DYNAMITE. <http://www.dynamite-project.org>
3. EAST-EEA. <http://www.east-eea.net>

4. EMBASSI. <http://www.embassi.de>
5. SMARTKOM. <http://www.smartkom.org>
6. SMARTWEB. <http://www.smartweb-projekt.de>
7. TOPPRAX. <http://www.topprax.de>
8. USEKIT. <http://www.usekit.de>
9. Fabbio Paternò: ConcurTaskTrees - An Engineered Notation for Task Models. In D. Diaper, N. Stanton (Eds.), The Handbook of Task Analysis for Human-Computer Interaction, pp.483-503, Lawrence Erlbaum Associates, Mahwah, 2003
10. Joaquin Miller, Jishnu Mukerji (Eds.): MDA Guide Version 1.0.1, Object Management Group, 2003, [URL:http://www.omg.org/docs/omg/03-06-01.pdf](http://www.omg.org/docs/omg/03-06-01.pdf)
11. MDA Guide Version 1.0.1, OMG, 2003. www.omg.org/docs/omg/03-06-01.pdf
12. OGM QVT: MOF2.0 Query/Views/Transformations RFP, <http://www.omg.org/docs/ad/02-04-10.pdf>
13. OMG MOF: Meta Object Facility Specification, <http://www.omg.org/technology/documents/formal/mof.htm>
14. OMG UML: Object Management Group Unified Modeling Language: Superstructure, October 2004
15. W3C CC/PP: Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0, W3C Recommendation, 15. 1. 2004, <http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115>
16. W3C EMMA: Extensible MultiModal Annotation markup language, W3C Working Draft, 14. 12. 2004, <http://www.w3.org/TR/emma>
17. W3C X+V: XHTML+Voice Profile 1.0, W3C Note, 21. 12. 2001, <http://www.w3.org/TR/xhtml+voice>