A MODEL BASED APPROACH FOR DEVELOPING ADAPTIVE MULTIMODAL INTERACTIVE SYSTEMS

Waltenegus Dargie, Anja Strunk
Technical University of Dresden, Faculty of Computer Science, Chair of Computer Networks

Matthias Winkler
SAP Research

Bernd Mrohs
Fraunhofer Institute for Open Communication Systems, Competence Center Smart Environments

Sunil Thakar, Wilfried Enkelmann
DaimlerChrysler AG, Group Research and Advanced Engineering

Keywords: Multimodality, model driven architecture, software engineering, context-aware computing transformation.

Abstract: Currently available mobile devices lack the flexibility and simplicity their users require of them. To start with, most of them rigidly offer impoverished, traditional interactive mechanisms which are not handy for mobile users. Those which are multimodal lack the grace to adapt to the current task and context of their users. Some of the reasons for such inflexibility are the cost, duration and complexity of developing adaptive multimodal interactive systems. This paper motivates and introduces a modelling and development platform – the EMODE platform – which enables the rapid development and deployment of adaptive and multimodal mobile interactive systems.

1 INTRODUCTION

At present, the way users interact with a plethora of devices while being mobile and/or carrying out other impending activities – such as driving, attending a child, holding a presentation, etc. – can be a potential challenge to the vision of ubiquitous computing. Besides the inconveniency of managing those devices, often interaction takes place via impoverished traditional input and output systems which require considerable attention and engagement.

Multimodal interactive systems significantly reduce the obtrusiveness mobile devices impose by increasing the communication bandwidth between the user and the devices with which he interacts. A gain in a communication bandwidth may offer flexibility and freedom, allowing the eyes to focus on and hands to be free to operate or to handle other objects while an interaction with a device takes place. Having said this, present multimodal interactive systems do not exhibit the grace to adapt to the social and conceptual settings in which they operate, both to offer optimal flexibility and to guaranty privacy.

Developing adaptive multimodal interactive systems is a costly and time consuming process. This is partly because of the inherent complexity of interactive systems, but also because of the absence of standardized methodology and integrating toolkits which can be employed during the development process. Within the EMODE project, we are tackling these issues by developing a methodology as well as a development and runtime support for multimodal adaptive interactive Systems. In this paper, we will give an overview of the EMODE platform and discus its methodology.

Dargie W., Strunk A., Winkler M., Mrohs B., Thakar S. and Enkelmann W. (2007).
A MODEL BASED APPROACH FOR DEVELOPING ADAPTIVE MULTIMODAL INTERACTIVE SYSTEMS.
In *Proceedings of the Second International Conference on Software and Data Technologies - PL/DPS/KE/WsMUSE*, pages 73-79
DOI: 10.5220/0001334300730079

The remaining part of this paper is organized as follows: in section 2, we will discuss related work; in section 3, we will introduce the EMODE methodology; in section 3, we will present the EMODE development environments; in section 4, we will introduce the In Car Adaptive Travel Assistant we are planning to develop by using the EMODE platform; and finally in section 5, we will give concluding remarks and outline future work.

2 RELATED WORK

Single authoring approaches enable service authors to create their services regardless of the way the services will be accessed (Mandyam, 2002); authors merely describe the user interfaces of their services, and an adaptation engine takes care of the transformation to specific target devices and platforms.

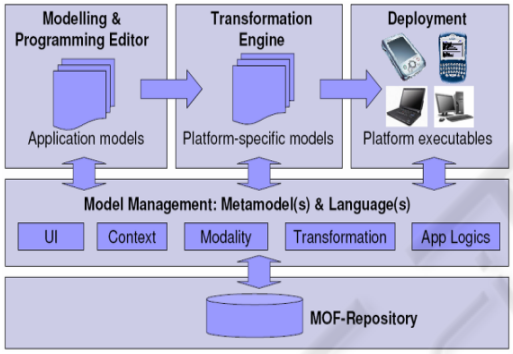

Figure 1: The EMODE Conceptual Architecture.

The ADAPT framework (Steglich, 2004) employs a device independent UI description language for applications to produce their output in a format called the Generic User Interaction Mark-up Language, an XML based user interface description, which will eventually be transformed into device specific output formats such as HTML, WAP or VoiceXML. ADAPT also supports the transformation of a user's input into application input parameters. The Renderer Independent Markup Language (W3, 2002) and the Device-independent Multimodal Markup Language (Hu 2005) provide similar support, but lack the support for graphical UI interfaces. The SISL (Ball, 2000) approach supports interaction based on natural-language processing, both for speech and text, and integrates those user interfaces with web and touch-tone interfaces on the basis of common service logic. While SISL describes how to realize a service logic that can be interfaced by different user interfaces, it does not specify how these user interfaces should automatically be generated from a common source.

The Device-Independent Web Engineering Framework (Kirda, 2002) creates flexible and extensible multi-device services, supporting the availability of web applications on resource-limited devices such as PDAs and mobile phones. It allows mobile access to web content and describes XML/XSL based techniques and software tools for web developers to implement services that are device independent. The disadvantage of DIWE is that it focuses on and limits itself to a web developers' view.

A model-based approach to user interface design is presented in (Grolaux, 2001), which is based on the Oz programming language, focusing on graphical user interfaces. Multimodality and adaptation of the UI to contextual changes, however, is not supported. Similarly, Dery-Pinna et al. (Dery-Pinna, 2004) present a development environment based on components. The UI is considered as a technical service of a business component just like security or persistence. The dialog between UI and business components is managed by an interaction/coordination service that allows the reconfiguration of components without modifying them. A UI component merging service handles dynamic assembly of corresponding UI components. Calvary et al. (Calvary, 2002) propose a reference process to enable the modelling of user interfaces that keep their usability in different settings. The changing settings include the hardware and software platform as well as environmental factors, such as noise and lighting. While the main focus of this work is the preservation of plasticity, the major focus in EMODE is to facilitate the development process of multimodal interactive systems by creating an integrated development and runtime platform to improve the cost efficiency of the entire creation process. In this way, the reference process provides a basis upon which the EMODE platform builds.

A work similar to ours is the Adaptive User Interface Generation framework proposed by Hanumansetty (Hanumansetty, 2004). This framework introduces three concepts: firstly, a formalisation for representing a context is introduced. Secondly, a user interface generation life cycle is studied and a context model is defined on top of a task model, to introduce contextual conditions into the user interface generation process. This is useful for the user interface designer to specify contextual requirements and the effect of

these requirements on the user interface. Thirdly, context-aware adaptation of user interfaces is achieved by mapping context specifications to various levels of user interface generation life cycle.

The work, however, does not discuss how modality and context are modelled and how model transformation takes place.

3 THE EMODE APPROACH

In the final analysis, the usefulness of an application is judged by how best it meets the needs of its user, regardless of the underlying technology. This makes it necessary to develop adaptive and flexible applications and interactive systems. This is of particular importance in a pervasive computing environment, as the context and activity of the user potentially change over time, making it infeasible to foresee at design time the type of devices, networks, platforms, and modalities involving an interaction.

The model-driven architecture approach meets this requirement, enabling the designer of an adaptive multimodal interactive system to model the application's logic along with other non-functional aspects such as the activities of the user and environmental settings without actually specifying any particular platform or device. Platform specific aspects and behaviours, such as the type of modalities suitable for a given interaction setting, device capability, display type, communication bandwidth, etc., can be described by a transformation policy which guides the mapping of the platform independent model into a platform dependent model, from which a part of the actual software code can eventually be produced. The semi-automation of the transformation of the platform independent model to a platform dependent model and the platform dependent model into a software code enables the efficient and fast development of adaptive multimodal interaction systems.

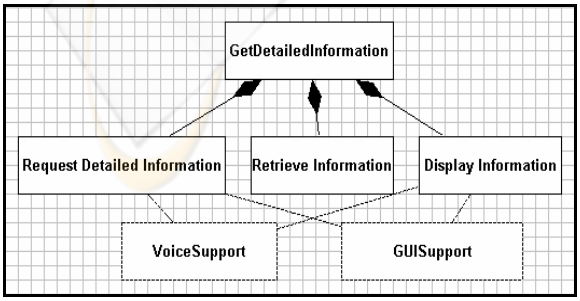

Figure 2: Sample Goal Model.

3.1 Methodology

An integrated approach is necessary for the development of adaptive multimodal interactive systems by keeping the focus on both issues: the user interface and the applications functionality. Several methodologies with focus on either software engineering or user interface development exist. They are, however, limited in that one field does not consider the concern of the other field. The EMODE methodology exploits the experiences learned both in software engineering and multimodal user interface design to facilitate the development of adaptive multimodal interaction systems. It also exploits the context of interaction (virtual and real environments, persons, devices, etc.) supporting runtime adaptation.

The EMODE methodology comprises three entities: a project team, a process model, and a conceptual architecture.

Our project team consists of users, developers, and designers. The user of the final application is the main source of domain knowledge and specific requirements information. The developers are responsible for developing the application logic and the designers deal with user interface development. Through an integrated development environment, these three parties are linked together.

Our process model defines a number of phases and related artefacts throughout the development process. These phases comprise the gathering of requirements, performing high-level goal modelling and model transformations and finally the implementation. During these phases a number of models describing different aspects of adaptive multimodal interactive systems are manually created using metamodels while other models are created by a model to model transformation process. We also support model to code transformation to partly enable the generation of code. Although transformations ease the development process, we do not aim at fully automating code generation. There will always be remaining work for refining the code.

The conceptual architecture describing the infrastructure and tool support for the development process is shown in Figure 1.

3.2 Architecture

The EMODE architecture consists of a repository, a suite of metamodels, a model editor, a transformation engine and a deployment environment.

Since we are following a model-based approach, a repository holding the model information forms the base of the conceptual architecture. The model management layer is represented by the EMODE meta-models and languages. These are needed to perform application modelling and transformations during the modelling. An overview on the meta-model and the transformations will be given in the following section. The modelling and programming infrastructure is built on top of the repository and the model management layer. It consists of a number of modelling editors which enable developers to model user and system tasks, abstract user interfaces, and context. A transformation engine transforms one model into another model, such as a goal model into a task model or an abstract user interface description into a concrete modality such as voice or graphic. The conceptual architecture is currently being implemented as the EMODE tool chain. A first prototypical version of the tool chain is currently being tested.

3.2.1 Metadata

To model different aspects of multimodal adaptive interactive systems, we define a suite of meta-models. These include the Goal model supporting the modelling of functional and non-functional goals and the Task model for modelling the overall task flow consisting of user, system, and interaction tasks. The Concept model is used to model data concepts for the application. The AUI model and Modality model provide the means for describing abstract user interfaces and modality profiles of devices that will finally render the modelled UIs. Also of importance are the Context model and the Functional Core Adapter model by which service calls to backend application logic are modelled. During the development process, models based on these meta-models are either created by the developer directly or through a transformation from an existing model.

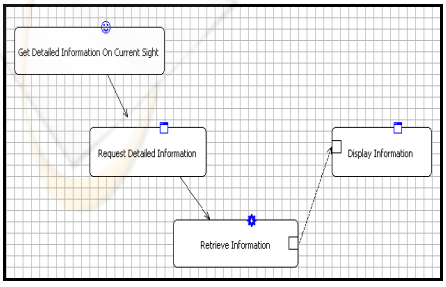

Figure 3: Sample Task Diagram with User Task, Interaction Tasks, and System Task.

To illustrate the usage of the meta-model and transformations, we will now provide a short example outlining the interplay of the different models.

At design time, a developer starts to model the goals of a new application from the user's perspective. Goals are divided into functional goals (the application's functionality) and non-functional goals (for example, the support of a speech modality for an interaction). Both types of goals are the outcome of a requirements-analysis phase. While the goal model is the entry point in the modelling process, the requirements-analysis phase is not supported by EMODE at present. Figure 2 demonstrates a user's (such as a tourist) intention to get detailed information about a particular sight. The intention is modelled as a higher-level goal. The application developer divides this goal into the sub-goals: "Request Detailed Information", "Retrieve Information", and "Display Information". While the sub-goal "Retrieve Information" refers to a goal accomplished by the application, the others represent interaction between the user and the system. Obviously, the higher-level goal entails non-functional goals related to interaction. An interaction can be carried out with the support of multiple modalities, for example, voice and GUI modality.

Each sub-goal will be translated into a more concrete task using concepts specific to the application. Subsequently, tasks are separated into the user's tasks, the system's task and an interaction task. A user's task describes the responsibility of the user to accomplish his goal; the system's tasks are those executed by the system regardless of the interaction type; and an interaction task is a task carried out by a modality service which is responsible for providing the necessary input for a system task and the necessary output to the user. A task diagram is generated semi-automatically by transforming the goal model.

Figure 3 displays a task diagram corresponding to the above goal diagram. A traveller triggers the information retrieval by interacting with the system; the system fetches the information, and displays it to the user in a manner consistent with the user's context.

In the next design phase, an abstract user interface is created by the AUI model editor. The AUI model is generated from the Task model and further refined by the developer. The transformation of the AUI model to code produces an adaptive, multimodal dialog description in the XML-based D3ML document format. In our approach, D3ML is utilized as an intermediate format which cannot directly be rendered but is transformed into a target

language such as VoiceXML or XHTML. Building modality-specific document formats from D3ML is a four phase process.

The first phase comprises the semantic adaptation of the D3ML dialog by evaluating inline annotations that contain conditions over context information. Semantic adaptation encompasses layout selection for each target modality with respect to the properties of the rendering device, for example, the screen size available to a visual modality output. Furthermore, UI controls whose context conditions are not satisfied are pruned from the dialog.

In the second phase, the previously selected modality-specific layouts are filled with content from the D3ML dialog, producing D3ML code that is structured according to the target modalities. This structure is the basis for the dialog fission phase that decomposes the D3ML dialog into separate adapted and modality-specific D3ML fragments.

Finally, the D3ML fragments are transformed into code understood by the particular renderers and recognizers of the target modalities. These transformations are purely syntactic mappings of mark-up code.

From a technological point of view, the first three phases are implemented by operations on the Document Object Model (DOM) tree of the D3ML dialog. For the transformations in the final phase, we pursue two approaches. The first approach utilizes declarative XSL transformation rules, which provide flexibility and easy extensibility, but come at the cost of low performance which makes them unsuitable on low-end devices. The second approach is an imperative implementation of the transformation rules over the D3ML DOM tree, which overcomes the performance issues of the XSL solution, but offers less flexibility. Despite this shortcoming, only the later allows us to perform D3ML adaptation on low-end, low-performance devices such as a PDA.

3.3 Development Environment

The EMODE development environment (figure 4) presents modelling tools and editors as well as transformation engines. The model editors integrate the EMODE meta-models and support the definition of various models, namely, goals, tasks, contexts, functional core adaptors, and various interactive modalities. The transformation engines provide the infrastructure support to specify and execute transformations. It includes, in more detail, a transformation language, a transformation editor, and a transformation front-end. The transformation front-end provides support for the user to control and prepare the execution of transformations. Apart from the above tools, our development environment integrates coding tools and management tools. The coding tools are used to complete, modify, compile and test the generated source code whereas the management tools are used to manage several repositories which are responsible for storing a large number of artefacts: models, transformations, source code and transformation traces. The model editors and the transformation front-end are developed as Eclipse 3.2 Plug-Ins.

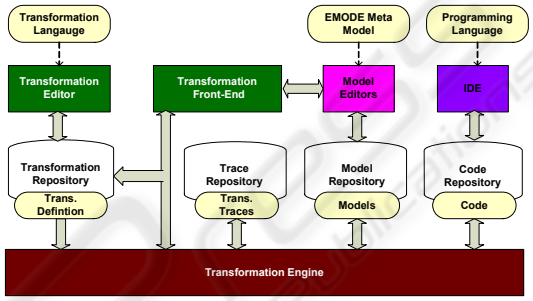

Figure 4: The EMODE development Environment.

3.3.1 Transformation

Transformations play an important role within the EMODE tool chain. They support the user while modelling an application by creating new models, which are derived from existing models. Thus, models can help to reduce the amount of work necessary to model applications as well as avoid modelling mistakes.

Two types of transformations were implemented within EMODE: model-to-model transformations and model-to-code transformations. The former ones are used to derive a task model from an existing goal model and an AUI model (Abstract User Interface) and FCA model (Functional Core Adapter) from a task model. These transformations are described in a declarative QVT syntax and executed by a prototypical QVT transformation engine.

Model-to-code transformations are used to generate a significant body of an application automatically. In most cases the developer will have to add additional code to implement the application logic. Within the EMODE project we implemented a transformation that generates Java AWT code using JET (Java Emitter Templates). A second transformation based on the XML Template Language (XTL) generates D3ML (Device-independent Multimodal Markup Language) dialogues.

EMODE applications comprise different parts which are generated from the different models. At present, the EMODE development environment supports the following model/code mappings:

- The concept model is used to generate the code that implements the data types used throughout the application;
- The code implementing the overall application workflow is generated from a task model;
- From the functional core adapter model method stubs are generated that make up the interface to the application's business logic;
- The abstract UI and modality model are the source for generating the nodes which describe the application's user interface; currently, we support two different transformation targets for UI: plain Java/AWT and D3ML. The latter will eventually be transformed into different modality-specific dialogue descriptions such as XHTML or VoiceXML; and,
- The context model is used to automatically generate context provider descriptions.

Model-to-code transformations in EMODE currently use Java SE as target platform. They can, however, easily be adapted to support other target platforms, such as .NET. The transformations are implemented using Java Emitter Templates.

4 FUTURE WORK

We are developing two demonstrators using the EMODE tool chain. The first demonstrator, the in-car travel assistant, is already developed with standard tools; our future aim is to produce it with the EMODE tool support. This will enable us to compare our methodology with existing standard approaches in terms of efficiency of developing adaptive multimodal interaction systems. The second demonstrator is a mobile maintenance application which provides support for maintenance workers to access relevant information and compile a report regarding their present task. In this paper, we will briefly introduce the in-car travel assistant only.

4.1 In-Car Travel Assistant

The in-car travel assistant is an adaptive multimodal interactive system. It provides personalised, situation-adapted information during a trip. The situation context can be acquired by different sensors and data sources. Some of these are data available in the vehicle over the CAN bus, such as speed, fuel tank level, outside and inside temperature, and the current geographical position. Additional sources of a context are the mobile devices available inside the car.

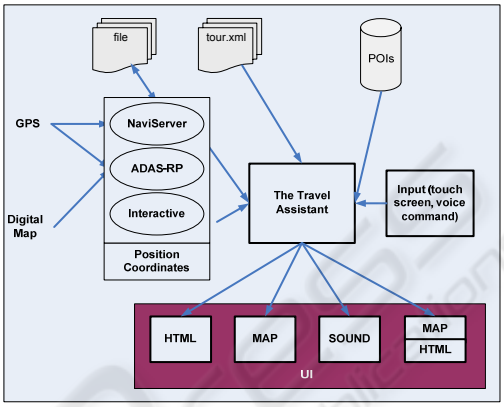

Figure 5: Architecture of the current in-car version of Travel Assistant.

The in-car travel assistant acquires and uses a context to support two aspects of adaptations: adaptation of the application's business logic and adaptation of the way information is delivered by and to the user, i.e., adaptation of the modalities employed. Adaptation of the application logic takes place when travel information is selected, filtered, and presented to the user based on the user's current situation (mostly location) and preferences. Modality adaptation takes place by combining different input and output modalities and operating possibilities depending on the user's situation. For example, to announce a point of interest by a speech modality while the user is talking on the mobile phone is irrelevant or even obtrusive; hence, the system selects an appropriate modality to present a point of interest less obtrusively. Additional constraints for the selection of appropriate modalities are privacy policy and environmental context (ambient noise and surrounding light intensity, etc.).

On-trip travel information (point of interest) is made available on portable devices of heterogeneous nature, such as a smart phone or a PDA or an embedded display device. At the same time, available multimodal services expose their services by advertising them so that the travel assistant can dynamically discover and bind to them.

The travel assistant is a device independent application. Depending on the preferred device or the context of use, it employs a web browser

(HTML) component to dynamically plan or alter trips; select points of interests and present information en-route to its users as text or image. It also possesses a map-based interaction mode. The architecture for the current version is shown in Figure 5.

In our current version, the travel assistant gets live position data (coordinates) either from the NaviServer, which in turn gets the live data from a GPS, or interactively by clicking on a map. NaviServer can also record and read the position data from a database containing previously recorded position coordinates. Additionally, the NaviServer can process the data coming from the ADAS-RP (Advanced Driver Assistance System – Research Platform), which is in itself connected to a GPS and typically provides map data, matched coordinates and additional information such as road types and road names.

The travel assistant has a link to POI-information system to provide the user all the detailed information regarding POIs selected during the pre-trip planning. As an output, the travel assistant, en-route a pre-planned trip, delivers information about the current position of a vehicle, POI-information and all the POIs in the vicinity of current position, in form of POI-view or Map-view or as combination of both (POI/Map) on a single screen.

5 SUMMARY

The EMODE methodology integrates software engineering concepts with user interface design approaches and adopts the model-driven architecture (MDA) to develop adaptive multimodal interactive systems. To define platform independent models and to support model-to-model as well as model-to-code transformation, we define a suite of meta-data for a goal model, a task model, a context model, a modality model and a functional core adaptor model. Whereas separating the platform independent aspects of an interactive system from those which are platform dependent enables a design time adaptation, the inclusion of context information throughout the modelling process of both the application business logic and the abstract user interface enables a runtime adaptation.

To quantitatively compare the EMODE methodology with existing standard approaches, we are developing two demonstrators: the in-car travel assistant and the mobile maintenance application. A first version of the travel assistant which is developed by standard tools is now fully functional; work is in progress to produce it with the EMODE tool chains, and to perform the comparison.

ACKNOWLEDGEMENTS

The EMODE project is partially funded by the German Federal Ministry of Education and Research (BMBF). We would like also to acknowledge the contribution of René Neumerkel and Gerald Hübsch to this paper.

REFERENCES

Mandyam, S., Vedati, K., Kuo, C., and Wang, W., 2002. User Interface Adaptations: Indispensable for Single Authoring. Position Paper of W3C Workshop on Device Independent Authoring Techniques, SAP University, St. Leon-Rot, Germany.

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

http://www.w3.org/2002/07/DIAT/posn/nokia-ibm-sap.html.

Hu, T.-Y., Park, D.–H., Moon, K.-D., 2005. Device-Independent Markup Language. Fourth Annual ACIS International Conference on Computer and Information Science (ICIS'05), 2005.

Ball T., and Sisl, 2000. Several Interfaces, Single Logic, Microsoft Research.

Kirda, Engin, 2005. Engineering Device-Independent Web Services, Ph. D. Thesis.

Delgado, R.L. and Araki, M., 2005. Spoken, Multilingual and Multimodal Dialogue Systems. John Wiley and Sons, Ltd.

Grolaux, D., Van Roy, P., Vanderdonckt, J., 2001. QTk: An Integrated Model-Based Approach to Designing Executable User Interfaces, Lecture Notes in Computer Science, vol. 2254, Springer Verlag.

Dery-Pinna, A. and Fierstone, J., 2004. User interface development environment based on components: assets for mobility. In Proceedings of the 1st French-Speaking Conference on Mobility and Ubiquity Computing. ACM Press, New York, NY.

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

Hanumansetty, R. 2004. Model Based Approach for Context Aware and Adaptive user Interface Generation, MSc. Thesis.