Towards Model-driven Development of Mobile Multimodal User Interfaces for Services

Daniel Porta
German Research Center for Artificial Intelligence (DFKI)
Campus D3_2
66123 Saarbrücken
daniel.porta@dfki.de

Abstract: We describe a model-driven approach for developing multimodal user interfaces for services that are especially appropriate for mobile consumption. Thereby, we refine an existing service engineering methodology with respective meta-models and provide initial tool support for user experience experts by means of an Eclipse-based, graphical editor implementing model-to-model transformations, consistency checks, and automatic code generation. The resulting runnable UI can either be consumed from a desktop pc as well as from modern smartphones via browser-based, multimodal client applications. This allows for a seamless user experience throughout different devices and situations. Such a client application is linked to a service-oriented, ontology-based dialogue platform that acts as a middleware between the client and the service backend and enables a user to naturally interact with services.

1 Introduction

Nowadays, the ubiquitous access to whatever information is indispensable. This ranges from comparably simple weather or traffic jam information which a user might want to access from at home, work, or on the go to rather complex business applications like workflow or enterprise resource planning systems. Mobile operating system platform providers created their own application distribution systems, which allow users to find, download and seamlessly install a vast amount of highly specialized applications on their mobile device. Although billions of iPhone applications have been downloaded so far, statistics[1] report that over 60 percent of paid applications have been pirated. As a solution, the Internet of Services enables innovative and competitive business models and allows for reducing costs. At the same time, intellectual property of service providers is better protected. Former monolithic blocks of applications are split up into functional services that can be composed and that communicate with each other over the Internet. For example, a service provider implements a business backend service and does not care about an appropriate user interface (UI). This, in turn, is implemented in a dedicated frontend service (that uses the aforementioned backend service in its composition chain) by another service provider who is an expert in the field of intelligent service UIs. For rendering such UIs only a thin client and a working Internet connection for data transmission are needed. Such a thin

[1] http://www.pinchmedia.com/blog/piracy-in-the-app-store-from-360idev/

client could, e.g., be a modern smartphone with a full-fledged Web browser. However, mobile devices still suffer from limitations that affect the usability of mobile applications. Here, natural interaction metaphors are especially feasible.

In this paper, we present our model-driven approach for providing multimodal UIs for services. These UIs can either be accessed from a desktop pc as well as from modern smartphones, since for rendering the graphical part of the UI, only a full-fledged Web browser is required. This allows for seamlessly providing users the same user experience throughout different devices and situations. The outline of the paper is as follows. Chapter 2 gives a short introduction to the conceptual framework underlying our approach for modelling multimodal service UIs that is described in chapter 3. A runtime environment for multimodal service consumption is discussed in chapter 4. Finally, we conclude in chapter 5.

2 Integrated Service Engineering

The Integrated Service Engineering (ISE) methodology [CVW08] ensures a proper service description for automatic composition, discovery, execution, and provisioning of the developed service. As illustrated in figure 1, the ISE methodology accounts for different service facets at different abstraction layers and hence maps to the typical workflow for developing an electronic service in larger organizations. At the highest business layer, the description of a service facet is very abstract, while with each lower layer, it becomes more concrete. Following the model-driven engineering paradigm, each intersection of a facet and a layer is represented by a specialized meta-model. Conceptually, there exist vertical model transformations for transforming an instance of a higher-level meta-model to a more concrete meta-model. Ideally, reverse transformations are also possible in order to allow for iterative model refinements. Dependencies between different service facets are resolved by horizontal model integrations. Based on the Eclipse Rich Client Platform (RCP), the ISE workbench [SVB+09] prototypically implements the ISE methodology. Developed in the TEXO[2] use case of the German THESEUS Research Programme, the workbench additionally implements a one-click deployment feature, which creates a service archive containing all defined models and automatically deploys it on the TEXO service platform. Important to mention is that, depending on authentication and authorization parameters, in principle all models can be retrieved from a service archive hosted on the platform.

So far, the main focus within the methodology laid upon the *Service* and *Workflow Descriptions*. We have been working with the proposed *UI Description* process. As it turned out, a link from the UI to the technical service is hard to establish. Based on the experience we gained while working with the methodology, we propose a refinement to the process in order to further allow for the development of multimodal service UIs. Affected matrix cells in figure 1 are highlighted in blue, proposed refinements to the methodology are written in a bold font and discussed in the next chapter.

[2] http://www.theseus-programm.de/en-US/home/default.aspx

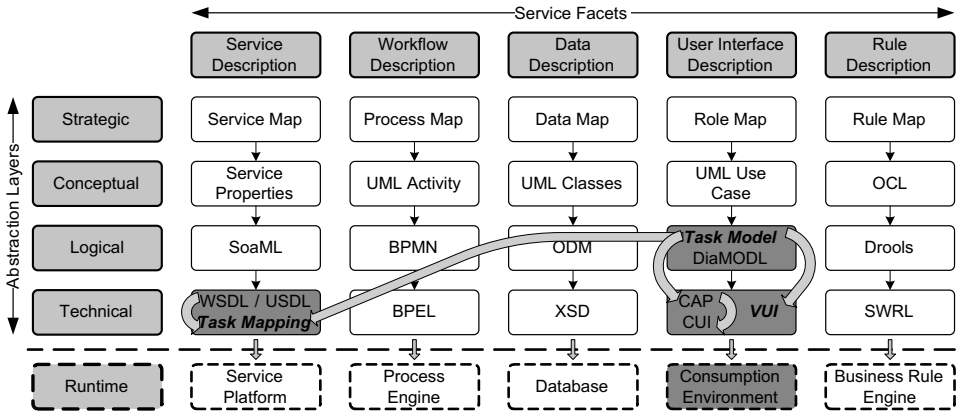

Figure 1: The Meta-Model Matrix of the ISE Methodology

3 Model-driven Development of Multimodal Service UIs

In this chapter, we describe the necessary meta-models and steps in the model-driven development for implementing executable multimodal service UIs. Usually, user interaction with business services is task-driven, meaning that a service supports a user in performing a certain task. Thereby, business services often implement business processes that involve multiple human actors. Such relations can be described on a very high-level by means of *UML Use Case* diagrams within the *Conceptual UI Description* matrix cell of the ISE Methodology. Unfortunately, expressiveness of the *UML Use Case* profile is restricted. For example, explicit sequences of use cases cannot be modelled. For this, the *UML Activity* profile that is also used for the internal *Conceptual Workflow Description* would be more appropriate. A *UML Activity* diagram can then (together with the preceding *UML Use Case* diagram)[3] be transformed into a fine-grained user-centric *Task Model* that accounts for hierarchical task decomposition and collaboration between different actors.

By means of a dedicated *Task Mapping Model*, the *Task Model* constitutes the abstraction layer between user intentions and the *Technical Service Description*. Furthermore, both meta-models are used by the service consumption runtime environment (chapter 4) in order to control the dialogue between the user and a service. The placement of the *Task Mapping Model* into the *Technical Service Description* matrix cell is based on the fact that *Technical Service Description* artifacts (i.e., a *WSDL* or *USDL* description) must exist before a task mapping can be defined. It turned out that, without this additional layer of abstraction, the immediate step from the *UML Use Case* diagram of the *Conceptual UI Description* to the *DiaMODL* [Træ06] model of the *Logical UI Description* does not leave enough space for a later technical service integration in the *Technical UI Description* matrix cell.

[3] The activity diagram does not contain information about involved actors. This information has to be derived from the use case diagram. Alternatively, the immediate transformation from the use case diagram to the task model is feasible.

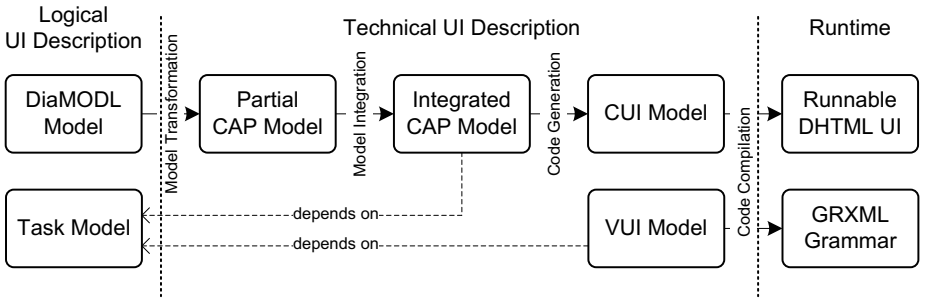

Figure 2: Overall model-driven UI Development Process

Figure 2 depicts a more detailed view on our proposed model-driven UI development process. Starting with a *DiaMODL* model, we provide a transformation to the partial abstract UI model (*CAP*). We denote this model partial, since it is not yet functional (if a user pushes a button on the GUI, nothing would happen at the backend). Functionality is weaved in with the integration of the *Task Model* (the dashed line with an arrow indicate a dependency of the integrated *CAP* model to the *Task Model* in UML notation). This results in the integrated *CAP* model that is eventually used for generating a concrete UI (*CUI*) model. Finally, the *CUI* model can be compiled at runtime into a DHTML user interface by means of the OpenLaszlo[4] compiler. In this context, the *CAP* model can be considered as platform-independent meta-model whereas the *CUI* model denotes the platform-dependent meta-model since it is tailored to the OpenLaszlo framework. Additionally, a language (*VUI*) model can be specified depending on the *Task Model*. This is then compiled at runtime into a GRXML-based speech recognizer grammar allowing the user for multimodal interaction with the accessed service.

The presented workflow is implemented by an initial set of Eclipse-based editor plugins and transformations that have been integrated in the ISE workbench. For defining a *VUI* model and a *Task (Mapping) Model*, we use a subset of editors contained in our own Eclipse-based workbench for building multimodal dialogue applications [SSNH09] that also allow for auto-completion and model validation. Hence, the development of these models is currently not fully integrated in the model-driven process, but technically integrated in the same working environment.

4 Service Consumption Runtime Environment

In order to accommodate the limited processing capabilities of mobile platforms, we utilize the Ontology-based Dialogue Platform (ODP) [PSN09] as a runtime environment for

[4]OpenLaszlo is an open-source Rich Internet Application (RIA) framework. It allows for rapid application development by means of a declarative XML-based programming language that can be compiled either into an Adobe Flash application or into DHTML that can be rendered by a bare Web browser without the need for a Flash player. http://www.openlaszlo.org/

service consumption. The ODP itself is implemented in a distributed, service-oriented architecture, where every major component can be run on a different host and communicate via HTTP/REST-based interfaces. This increases the reliability and robustness of the overall system. The major components comprise one or more (not necessary mobile) clients, a dedicated service for automatic speech recognition (ASR) and natural language understanding (NLU), a text-to-speech (TTS) service, and a dialogue system with generic access to a service backend.

In general, the *client application* is, according to our notion, designed as a lightweight component. It consists of a full-screen Web browser and a native, platform-dependent audio streaming library for sending and receiving real-time audio data. Currently, implementations are available for the iPhone platform and the Android platform. Thereby, a uniform client-side JavaScript API handles the communication with the native audio streaming part and the REST-based endpoint of the dialogue system. Desktop web browsers can (in combination with an optional java applet for audio streaming) of course also render the DHTML-based service GUIs, which eases the development process.

The *dialogue system* acts as middleware between connected clients and the service backend. Among other things, it provides a runtime environment for multimodal dialogue applications supporting advanced dialogical interaction based on domain-specific models. The models describe the reaction to pointing gestures and the representation of displayed graphics (GUI model), the natural language understanding process (language model), and the speech output. It also provides means for retrieving relevant models from a service archive and for invoking service operations due to a proper service description. The *CUI* model of a service is transformed into an internal semantic representation of the GUI and at the same time compiled into DHTML for rendering on the (mobile) client. The *VUI* model of a service is compiled into the appropriate GRXML-based speech recognizer grammar and delivered to the ASR/NLU service via the REST-interface. The *Task Model* is used by the dialogue manager (the central component of the dialogue system) that is responsible for controlling the dialogue with the user. It abstracts from concrete UI elements and solely operates on task description. Moreover, the dialogue manager initiates service calls at the service backend for which the *Task Mapping Model* is required.

The *service backend* consists of one or more service platforms that host different services. Such a service is bundled in a service archive containing all models that were created during the model-driven development. The dialogue system is granted access to relevant models. Furthermore a service can be invoked from the dialogue system. Subsequently, the dialogue system receives a respond from the service and presents it to requesting client.

5 Conclusion & Future Work

We consider natural user interaction with electronic services as an important success factor for the Internet of Services. We presented a model-driven approach towards the development of multimodal user interfaces for services. Thereby, we proposed a refinement of the underlying ISE Methodology. Most notably here is the introduction of a fine-grained *Task*

Model and a *Task Mapping Model* that are intended to close the gap between the user's intentions and resulting technical service calls in the backend. The development process is supported by a set of Eclipse-based editors that have been integrated in the ISE workbench. However, the creation of a *VUI* and a *Task (Mapping) Model* is currently not fully integrated in the overall process. The technical implementation of this has surely to be done as future work. A runtime environment for multimodal (mobile) service consumption consisting of a lightweight (mobile) client and a dialogue system that acts as a middleware between the user and the service backend allows for advanced dialogical interaction with services. This is especially feasible in a mobile situation, where users are often distracted by an eye-busy primary task, hence reducing the cognitive load of a user.

6 Acknowlegdments

This research has been supported by the THESEUS Research Programme in the Core Technology Cluster WP4 and the TEXO use case, which was funded by means of the German Federal Ministry of Economy and Technology (01MQ07012). The authors take the responsibility for the contents.

References

[CVW08] Jorge Cardoso, Konrad Voigt, and Matthias Winkler. Service Engineering for the Internet of Services. In Joaquim Filipe and José Cordeiro, editors, *Enterprise Information Systems, 10th International Conference, ICEIS 2008, Barcelona, Spain, June 12-16, 2008, Revised Selected Papers*, volume 19 of *Lecture Notes in Business Information Processing*, pages 15–27. Springer, 2008.

[PSN09] Daniel Porta, Daniel Sonntag, and Robert Neßelrath. A Multimodal Mobile B2B Dialogue Interface on the iPhone. In *Proceedings of the 4th Workshop on Speech in Mobile and Pervasive Environments (SiMPE '09) in conjunction with MobileHCI '09*. ACM, 2009.

[SSNH09] Daniel Sonntag, Gerhard Sonnenberg, Robert Nesselrath, and Gerd Herzog. Supporting a Rapid Dialogue Engineering Process. In *Proceedings of the First International Workshop On Spoken Dialogue Systems Technology (IWSDS)*, 2009.

[SVB+09] Gregor Scheithauer, Konrad Voigt, Veli Bicer, Matthias Heinrich, Anja Strunk, and Matthias Winkler. Integrated service engineering workbench: service engineering for digital ecosystems. In Richard Chbeir, Youakim Badr, Epaminondas Kapetanios, and Agma J. M. Traina, editors, *MEDES '09: International ACM Conference on Management of Emergent Digital EcoSystems, Lyon, France, October 27-30, 2009*, pages 446–449. ACM, 2009.

[Træ06] Hallvard Trætteberg. A Hybrid Tool For User Interface Modeling And Prototyping. In Gaëlle Calvary, Costin Pribeanu, Giuseppe Santucci, and Jean Vanderdonckt, editors, *Computer-Aided Design Of User Interfaces V, Proceedings of the Sixth International Conference on Computer-Aided Design of User Interfaces, CADUI 2006 6-8 June 2006, Bucharest, Romania*, pages 215–230. Springer, 2006.