Location-Transparent User Interaction for Heterogenous Environments

Chris Vandervelpen, Kris Luyten and Karin Coninx

Expertisecentrum Digitale Media – Limburgs Universitair Centrum
Wetenschapspark 2
B-3590 Diepenbeek – Belgium
{chris.vandervelpen,kris.luyten,karin.coninx}@luc.ac.be

Abstract

Because of the rapid evolution of mobile and embedded computing devices, it is important for a business that its services are accessible for a large group of possible service consumers. These consumers want to gain access to the offered services from heterogeneous environments. In this context, much work has been done on the (semi-) automatic generation of user interfaces (UIs) for services that are mostly targeted toward mobile computing devices. However, most of that work fails to provide actual solutions to the problem of how to provide a suitable interaction mechanism between the generated UI and the application logic that implements the functionality. We will show that by defining the implementation of the application logic as a webservice, we can use webservice technologies to provide two-way communication between the application and its generated UI.

1 Introduction

In a time that large quantities of new networked mobile and embedded devices are shipped to market, it is very important for businesses to make their services available for all these different kinds of appliances (e.g. PDA, cellular phone, set top box, information kiosk, desktop,…) with a short time-to-market. The main problem in achieving this goal is the diversity of all these devices. They all have different limitations such as different operating systems, memory sizes, screen sizes and interaction possibilities (touchscreen, stylus, speech, mouse, keyboard,…). When porting a UI from one environment to another, traditional development techniques require a redesign of the UI in order to take particular constraints into account. To address this problem, much work has been done concerning the (semi-) automatic generation and adaptation of UIs on different hardware and OS platforms (Eisenstein, Vanderdonckt & Puerta, 2001, Nichols et al., 2002)**.** They provide different ways to describe a UI on a platform independent level. This abstract platform independent description then should be instantiated with a concrete platform dependent UI by using a rendering mechanism (e.g. a web browser that renders HTML content). These approaches are often model-based: different models are used that together describe a complete UI. E.g.: A presentation model is used to describe the presentation structure, a domain model can be used to describe problem domain concepts and a platform model is used for describing the possibilities of the target platform. However, considerable less attention has been paid to the way one copes with two-way interaction between the generated UI and the implementation of its functionality which could be either a remote or a local application.

This paper extends previous work (Luyten, Vandervelpen & Coninx, 2002) where a presentation model and the UI rendering framework Dygimes (Dynamic Generation of Interfaces for Mobile and Embedded Systems) are used to generate UIs for heterogeneous computing environments. The Dygimes framework is a platform, device and system independent rendering and interaction system implemented in Java that uses XML-based abstract UI descriptions to represent the presentation model.

In the next section we will discuss the main problems with interaction when dealing with (semi-) automatic generated UIs for local and remote applications. Then, we introduce a solution for these problems based on webservice technologies. In section 5 we discuss benefits and shortcomings of our approach together with some ideas we like to work on in the near future.

2 The need for an interaction model

In general, a UI has to enable the user to interact with a piece of application logic and to let that application logic provide feedback (visual, sound, haptic,…). To enable this kind of two-way communication, UIs are tightly coupled with the application logic. However, when the UI and the application logic are not located on the same device, the tight coupling between both causes problems. Elaborating on a model-based UI design, we propose the usage of an interaction model to overcome the high coupling problem. An interaction model will help the designer to prepare the service or application logic to use location transparent UIs. One can think of this as an "enhanced" Model-View-Controller (MVC) architecture which will be necessary for the next generation of UI toolkits. Such a model is needed to enable the generated interface to be loosely coupled with the application logic.

Until now the Dygimes framework supported only a basic notion of interaction handling where action elements in the abstract UI description could be attached to an interactor. Such an action element denotes which method, from which Java class, would be invoked when the user manipulates that interactor. The problems with this approach were the restrictions to local method invocation and the binding to the Java programming language. Another problem was that there was no notion of datatyping in the system which was problematic when we wanted to pass data between the generated UI and the application logic. It is clear that these shortcomings restricted the interaction model.

A possible way to deal with the aforementioned problems would be to define our own communication protocol between the UI and the logic. For example, (Nichols et al., 2002) uses an XML-based communication protocol that enables two-way communication between appliances and their Personal Universal Controller (PUC). However, they define a static set of self-defined messages and they do not provide a way to describe the interactions and the data flows supported by the system separately. This means that the UI designer still needs a good understanding of the application logic.

We propose a solution that combines existing message passing and communication techniques with out Dygimes framework. This eases the task for the UI designer and allows him to concentrate on designing UIs for multiple platforms (the view) instead of being bothered by the technical details imposed by the implementation and the integration of the application logic (the model). On the other hand, the application logic programmer can implement the services without having to worry about the target devices. Our approach allows a smooth integration of application logic and UIs at runtime, while allowing to separate them during development.

3 Interacting with webservices

The proposed solution is based on the believe that one needs a device and programming language independent description of the possible interactions with the system. This enables the UI designer and the application designer to do their work independently of each other. Such a functionality description supports the generated UI to decide which functionality to invoke as a response to certain user interactions.

To realize this, we define a piece of application logic as being a webservice and the generated UI as being the client that wants to use the webservice. With such an approach we can use existing webservice technologies to achieve our previously stated goals. In this context, we have chosen to use a functionality description based on the Web Services Description Language (WSDL, 2001). WSDL is an XML-based language that enables the description of the operations, messages and datatypes that are supported by a webservice.. Operations consist of request and/or response messages. The language uses some default primitive datatypes, which can be used as parameters and response values for messages (double, integer, boolean,….). It also provides a section to define custom datatypes. By default XMLSchema is used for defining datatypes. We extended this specification by adding a section in which we bind UI interactions to service operations. From now on we will refer to this WSDL-based description as the interaction description.

Once we have an interaction description representing the service functionalities and the bindings with the abstract UI, we can use this information to invoke operations when the user interacts with the generated UI. For this we need a protocol for handling the interactions as described in the interaction description. In our approach we use existing XML messaging protocols to achieve this. One such protocol is the Simple Object Access Protocol (SOAP, 2000). This protocol enables us to invoke functionality on webservices by using Remote Procedure Calling (RPC) based on an XML syntax. Another, more efficient implementation of XML-based RPC, is XML-RPC (XML-RPC, 1999). It is in our intention to evaluate and integrate both implementations into the Dygimes framework.

4 Interaction in Dygimes

With the discussions from the previous sections in mind, we extended the Dygimes framework with a more advanced interaction engine. We now can identify the following parts:

- A piece of application logic (the service) that is annotated with a description of its UI and a WSDL-based interaction description that denotes the functionality. The interaction description also provides a binding between the service and the UI;
- A rendering system which renders the UI description on a particular platform. At the moment, the possible rendering platforms include AWT, Swing, HTML and Applet. The rendering on a cellular phone and a PDA is supported through the J2ME CLDC (Java 2 Micro Edition);
- An interaction engine which uses the interaction description to automatically determine which functionality to invoke on the application logic (which is local or remote) when the user interacts with the generated UI.

We now describe the sequence of events between the different parts when the Dygimes framework is used. To start, the application logic is asked to send its UI and interaction descriptions. When Dygimes receives this XML documemts, it renders the UI and parses the interaction description to build a datastructure in memory. When the user interacts with a particular UI interactor, which has a unique identifier, the following steps are processed by the interaction engine of Dygimes:

- Based on the interaction description, operations triggered by the manipulation of the interactor are determined;
- Data needed for the operation invocation are extracted from the user interface. The datatypes information provided by the interaction description is used to determine the datatypes of the message parameters;
- The interaction engine uses XML-based RPC or Direct Method Invocation (DMI) to invoke the necessary methods of the service;
- The service invokes the method, returns a message back to the UI client and the UI is updated if necessary.

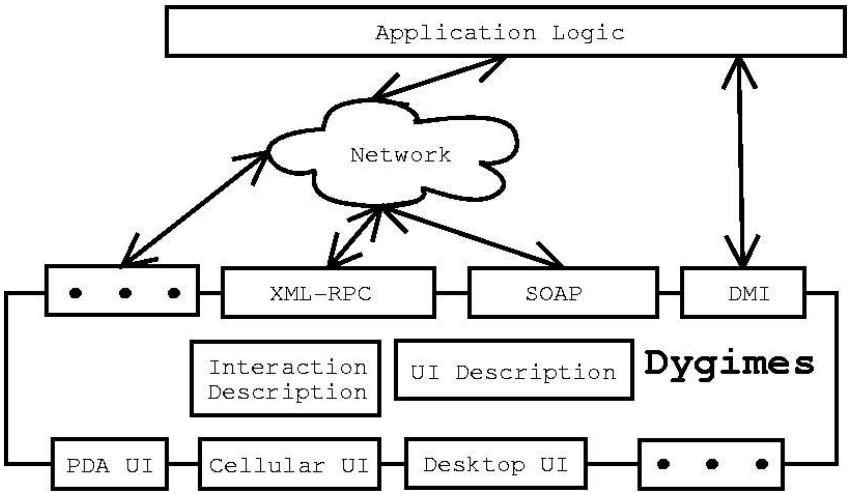

Figure 1: Architecture

Figure 1 gives a closer view of the architecture of the system. Dygimes uses the UI description to render the UI to heterogeneous environments. Interaction occurs through the use of the interaction description together with one of the communication technologies.

5 Conclusions and future work

In this paper we presented an extension of the Dygimes framework which enables the use of an interaction model to handle interaction. Because the proposed approach is based on existing webservices technologies, we can identify the following benefits:

- The system becomes webservices-enabled through the use of SOAP. This will be important in the near future;
- The approach is device and language independent because it is based on XML. If we for example used Java RMI, the system was restricted to the Java programming language;
- The system is usable through firewalls. This would not be possible if we used technologies like CORBA or RMI because they both use a binary communication protocol;
- The system uses common standards: XML and webservices;
- The generation of functional UIs for remote applications becomes much easier;

- The system becomes distributed which enables location-transparency. This means that the location of the application logic is transparent for the user which allows UI migration without the need to reconfigure the UI.

The main problem of this approach could be the overhead caused by the construction of the XML messages. For remote interaction this overhead is minimized by offering the user the possibility to use XML-RPC instead of SOAP. For local interaction we give the user a choice to use XML-based user interaction or DMI which is much faster. The drawback is that with DMI the user is restricted to Java implemented local services.

In the near future we would like to further investigate how webservice and XML technologies can help in achieving effective platform independent and multimodal interactive applications. We could, for example, use the definitions of datatypes in XML to make Dygimes extensible with new and composite interactors.

6 Acknowledgements

The research at the Expertise Centre for Digital Media (LUC) is partly funded by the Flemish government and EFRO (European Fund for Regional Development). The SEESCOA project (Software Engineering for Embedded Systems using a Component-Oriented Approach) IWT 980374 is directly funded by the IWT (Institute for the Promotion of Innovation by Science and Technology in Flanders).

References

Eisenstein J., Vanderdonckt J., & Puerta A. R. (2001). Applying Model-Based Techniques to the Development of UIs for Mobile Computers. In IUI 2001 International Conference on Intelligent User Interfaces, 69-76

Luyten K., Vandervelpen C., & Coninx K. (2002). Migratable User Interface Desciptions in Component-Based Development. In Forbrig P. et al., (Eds.), DSV-IS, volume 2221 of Lecture Notes in Computer Science, Springer.

Java 2 Platform Micro Edition. Sun Microsystems, http://java.sun.com/j2me/.

Nichols J., Myers B. A., Higgins M., Hughes J., Harris T. K., Rosenfeld R., & Pignol M. (2002). Generating remote control interfaces for complex appliances. In User Interface Software and Technology.

World Wide Web Consortium. (2001). Web Services Description Language specification. http://www.w3.org/TR/wsdl

World Wide Web Consortium (2000). Simple Object Access Protocol. http://www.w3.org/2000/xp/Group/

XML-RPC homepage. (1999). http://www.xmlrpc.com