DEVELOPING AGILE USER INTERFACES FOR HETEROGENEOUS DEVICES IN BUSINESS PROCESSES

Yaojin Yang and Lasse Pajunen
Nokia Research Center, Nokia Corporation, P.O.Box 407, FI-00045 Nokia Group, Finland

Keywords: Mobile device, business process, Business Process Execution Language for Web Services (BPEL/BPEL4WS), user interface development, Web Services, Service Oriented Architecture.

Abstract: Thanks to the increasing popularity of mobile devices, for accessing business process powered services people now can use various devices for different circumstances or tasks in order to have optimized performance. To support this kind of heterogeneous situation, user interfaces must be agile enough to adapt. In our research, we have identified five key requirements and five design guidelines to help developers to achieve this. Furthermore, we have introduced the concepts of user interface process and user interface service, where user interface development for business processes is well positioned in a bigger picture of Web Service and Service Oriented Architecture. Our research results have been presented by a case study developing a group messaging system.

1 INTRODUCTION

Supporting user interface agility, which is an ability to accommodate present and future changes on user interfaces, is becoming more crucial for developing successful business processes. In business processes, people and computer systems co-operate to achieve common business goals. Due to the increasing popularity of mobile devices, people tend to use heterogeneous devices for executing their tasks. Each business process usually involves multiple people. Therefore, the types of devices used in business processes are becoming quite extensive. Every participating device requires an appropriate user interface to provide as good as possible user experience. In addition, a business process usually involves more than one role. Each of these roles has its own interaction logic for interactions between a user and the process. Therefore, each role requires its own user interface. The extensibility and multiplicity of devices and business roles make the support of user interface agility an important requirement in developing business processes.

Offering a system or framework for implementing user interfaces for business process environments has been a topic of many research projects. PerCollab (Chakraborty, 2004) is a system that integrates business processes and various user interface technologies. In that system, an interaction controller manages all interaction with users. This approach puts much weight on the interaction controller. Therefore, it is suitable when a user interaction is simple, a basic request (for example filling in one form) from the user. However, we see that more complex user interaction is often needed. Other similar systems or frameworks have also been made. GreenBSN (Liang, 2005) and WOSE workflow framework (Lican, 2005) are some of the examples, where a gateway is used to provide a most suitable mobile interface for devices and to find a most suitable service for a certain situation. In Lynx (Velez, 2005), user interaction is implemented on top of an email system.

However, our research has been concentrating more on architectural issues of supporting user interface agility rather than developing specific techniques or systems. (Van Gurp, 2006) have also been working on a rather similar field. But, they are focusing on non-functional architectural requirements in general. Our research results include identified requirements for developing user interface for heterogeneous devices enabled business processes and proposed design guidelines that can be used in designing such systems. Particularly, we have introduced concepts of user interface process and user interface service. The user interface process is responsible for generating particular views for a

Yang Y. and Pajunen L. (2007).
DEVELOPING AGILE USER INTERFACES FOR HETEROGENEOUS DEVICES IN BUSINESS PROCESSES.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - SAIC*, pages 179-184
DOI: 10.5220/0002379401790184

certain user interface. It runs in parallel with business processes. In a Web Service enabled environment, functions implemented by the user interface process are published as a user interface service, which is a first-class service. We verified and evaluated the results by applying them to a group messaging case.

The rest of the paper is organized as follows. In Section 2, we will analyze typical situations in heterogeneous devices enabled business processes and will present requirements for suitable solutions. In Section 3, we will propose a set of guidelines that could be applied to a design and to an implementation of the design. In Section 4, we will use and analyze these guidelines in a group messaging case study. Section 5 will identify some future research topics, and finally in Section 6, we will draw conclusions.

2 REQUIREMENTS

In heterogeneous devices enabled business processes, user interfaces tend to change. A desktop is no longer the only device for a user. Instead, the interaction resource now consists of a dynamic set of devices including both desktops and mobile devices.

A mobile device has different user *interface capabilities* compared with a desktop and can also have different capabilities compared with other mobile devices in different models. Even the capacities of the same device can vary when it is operating in different context. The user interface capabilities includes hardware capabilities such as screen size, communication bandwidth, input mechanisms etc., and software capabilities like a browser capability, support of communication protocols etc.

In addition, mobile devices introduce push *style interaction*, where information is automatically delivered to users instead of being manually retrieved. This new type of interaction requires customizing user interfaces to have different interaction logic. On the desktop side, email based applications have similar situation. Therefore, user interface logic should be aware of the underlying interaction model and also be adapt to that.

Furthermore, in business processes there are usually more than one *role* in existence for interaction between a user and a process. Each role has its own interaction logic and one user could act in one or more of these roles. That is, each role has its own user interface and the user interface provision needs to be role based.

By analyzing mobilized business processes, we have discovered following requirements for supporting user interface agility in developing the heterogeneous devices enabled business processes.

R1: Providing customized user interfaces for each type of devices with a unique set of capabilities participating in business processes. This enables users using devices with different capabilities to interact with the same service provided by the business process.

R2: Providing customized user interfaces based on an available communication channel. This enables users in any context to access the same services using the same device. Switching between different user interfaces should be supported when the communication channel has been changed. The selection could be made by users or automatically by a system. For example, user interfaces for a Web browser is provided when a high-speed channel is available. However, when a high-speed channel is not available, SMS based user interfaces can still be provided to access the service. In addition, off-line user interfaces could be offered where no connectivity is available or allowed.

R3: Providing new customized user interfaces for new types of devices after business processes have already been deployed. When new types of devices are used, users should be able to use them to participate in processes immediately with a minimum user effort required. Also, there should not be any effect to the existing and running system.

R4: Supporting both pull and push approaches with different interaction logic implemented in user interfaces.

R5: Providing user interfaces based on users' roles in business processes. Different roles conduct interaction with services in different logics. An appropriate user interface should be provided once a certain role is initialized. Switching between different user interfaces should also be supported when the user's role has been changed.

3 DESIGN GUIDELINES

Based on the requirements above, we can say that a system of heterogeneous device enabled business processes should have a clear separation between its user interface and business logic. Then, user interface variation will not have any impact on business logic. In addition, the system should have a clear separation between its user interface presentation and data that is going to be presented in order to facilitate the user interface customization.

Furthermore, the system should offer a flexible and standard based deployment mechanism, so that it is able to dispatch role based user interfaces or customized user interfaces for new types of devices after deployment time.

Finally, because the system aims for providing and executing business process based services, it is straightforward to implement it as a Web Service based system, which secures the system to be an open and standard-based environment.

Hence, we have outlined the following guidelines that we should take into account when designing such systems. However, the outlined guidelines do not aim at providing specific techniques for satisfying the identified requirements. Instead, the guidelines aim at building a framework for facilitating the development that is going to realize the requirements. As a result, there is no one-to-one mapping between the guideline and the requirement.

G1: Model-View-Controller (MVC) is the general design pattern for designing a system of heterogeneous device enabled business processes, because of its supporting of loose coupling between user interface components and business processes.

G2: In the MVC model, View module needs to be further decomposed into the components of presentation and the components of view generation. Each device has its own user interface infrastructure. The components of presentation are part of that infrastructure; the components of view generation are independent of that infrastructure and responsible for generating views. The component of view generation is the place where the user interface customization is realized. Customization includes both presentation customization and the user interface logic customization. The set of components for generating views for certain customized user interface could be a stand alone deployment unit. For instance, a set of components responsible for generating HTML views and a set of components responsible for generating XForms views are two deployment units. Inside a View module, a client-server structure and messaging based communication can be adopted as design patterns for achieving loose coupling between presentation and view generation. This is necessary if the components of view generation are deployed on a server side instead of on a device side, which is a similar case compared to Web application development.

G3: Model module consists of business processes that control dynamic nature of the data, which are independent of user interfaces.

G4: Controller module deals with user inputs and manages view generation components according to the results of service invocation, and it also takes care of interpreting user inputs, determining which service to invoke, and then invoking that service. It is used as a communication hub between the view module and the model module.

G5: There are three types of component-level communication in existence in the system: device internal communication, server-side internal communication and cross boundary communication. The device internal communication makes use of the device's native communication infrastructure for a better performance. The server-side internal communication adopts standard messaging mechanisms in order to achieve maximum interoperability. Cross boundary communication should adopt a standard mechanism if possible. A bottleneck may be lacking support for the standard mechanism from mobile devices. Therefore, it might need to have adaptation components for enabling the standard-based device-to-server communication.

4 CASE STUDY

4.1 Introduction

We have designed and implemented a system of a group messaging process by taking the design guidelines into account.

In the group messaging process, a person can send messages to a group of people, and any person on the recipient list can receive and read the message and then send acknowledgement to the sender. The messages can be either plain textual messages or textual messages with binary attachments. One typical scenario is that (Figure 1). Firstly, all users who are going to send and receive group messages need to join the process and register their devices. Then, the sender composes a message, and sends the message. The system delivers the message to a group of recipients. The delivery can be performed in either push or pull style. It is decided by the specific type of device that the recipient is using. Once the recipients receive the message, they read it and then send acknowledgement to the sender. The system delivers the acknowledgement to the sender in the same way of the message delivery. The sender then checks the acknowledgement status from all recipients. If acknowledgement has already been received from all recipients, the sender tells the system to delete the message. If not, the sender continues to wait for new incoming acknowledgement. Finally, all users quit the group messaging process.

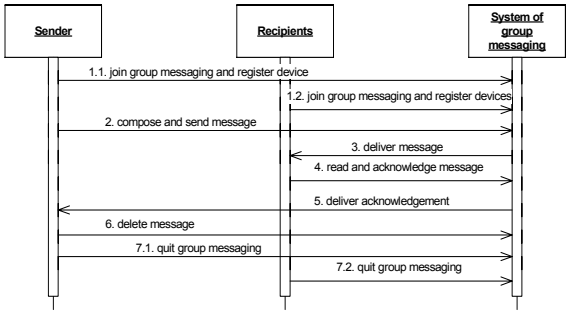

Figure 1: Group messaging sequence chart.

In this case study, there are three different types of devices that are going to be supported: desktop computers, high-end mobile devices with a decent HTML browser, and low-end mobile devices with SMS as the only messaging application.

4.2 Design

The design of the system is based on the framework exposed by the guidelines (Figure 2). The system is an application of the MVC pattern (G1).

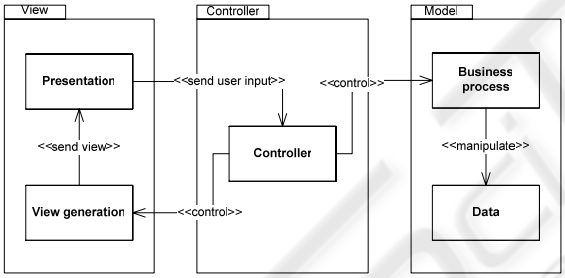

Figure 2: High-level design of the system of group messaging process.

In the View module, the class of Presentation represents a collection of related components for rendering device specific user interfaces; the class of View generation is a collection of related components for creating user interface views that are then going to be presented locally in each device (G2). Multiple instances of View generation will be deployed onto the system. Each of them will be deployed separately as a stand alone deployment unit and will take care of one certain type of user interface. Based on the types of devices that are going to be used in this case study, desktop computers, high-end mobiles with HTML browser and low-end mobiles with SMS support only, we decided to support two different types of user interfaces, HTML-based and SMS-based. The HTML-based user interface will serve both desktop computers and high-end mobiles with pull style interaction; the SMS-based user interface will serve low-end mobiles with push style interaction. Consequently, there will be two instances of View generation deployed separately onto the system. The instances could be deployed on device if the device's capability allows, but, in our cases, they will be deployed on server in order to achieve better performance and impose centralized control on view generation.

The Controller module serves as a communication hub between the View module and the Model module (G3). It will take user inputs from user interfaces, then interpret the inputs to requests of invoking certain operations on a data model, and finally manage the view generation to produce new views of the user interface according to results from operations.

In the Model module, the class of Business process represents a set of related components that implements group messaging related functions listed in Section 4.1; the class of Data encapsulates the data repository of the system and any data-related lower-level services for accessing and manipulating raw data (G4). In a message delivery procedure, the message is firstly sent to and saved in the data repository, and then retrieved from the data repository and sent to recipients.

As proposed by the last guideline (G5), the server side and device-to-server communication will be based on a standard mechanism, while the device side communication will be performed in the devices' native ways. About the standard mechanism, we naturally choose the Simple Object Access Protocol (SOAP) over other protocols as the primary communication protocol, because it is a de factor standard. However, since lacking SOAP support from mobile devices or considering performance issues, it is necessary to have additional adapters to convert protocols between HTML and SOAP and between SMS and SOAP.

4.3 Implementation

Figure 3 is the actual implementation of the system, which takes advantage of several technologies.

We use the Business Process Execution Language for Web Services (BPEL/BPEL4WS) (Curbera, 2003) as the implementation language of our processes due to several reasons. Firstly, our system is a business process-oriented system and all tasks can be modeled and implemented as processes. Secondly, the system is a long-running system,

which especially requires compensation actions and scoping to support failure recovery.

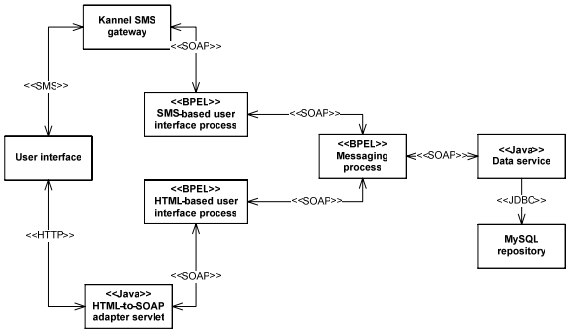

Figure 3: Implementation of the system of group messaging process.

Thirdly, the system needs to rely on a flexible deployment mechanism in order to be able to deploy components even after initial deployment. Last but not least, the system is a Web service based system.

Due to the flexible deployment mechanism supported by BPEL implementations, we decide to implement the instances of view generation, presented in the design, as BPEL processes as well and call those processes user interface processes. Therefore, in the implemented system there are three BPEL processes, *HTML-based user interface process*, *SMS-based user interface process*, and *messaging process*. The first two processes are responsible for view generation and also act as Controller role to connect the device to the messaging process. The last process is where the actual messaging related business logic is implemented. We use ActiveBPEL BPEL implementation to execute these three processes.

The HTML-to-SOAP adapter is implemented as a Java Servlet, which can convert messages between HTTP and SOAP. The SMS-to-SOAP adapter is based on the Kannel, an open source SMS gateway, which can convert messages between SMS and SOAP.

Relational database, MySQL, is used as the Data repository, which the messaging process can access to through the Data service implemented on Java Hibernate.

Table presents realization relations from the implementation component to the logical component in design, and deployed-onto relations from the implementation component to the deployment unit.

Table 1: Mapping table between implementation and logical components, and between implementation component and deployment unit.

Implementation component	Logical component	Deployment unit
User interface processes	View generation and Controller	BPEL engine on server side or user device side
Group messaging process	Business process	BPEL engine on server side
Data service	Data	Java runtime on server side
Data repository	Data	Database on server side
Communication adapter		Java Servlet container on server side

4.4 Experiences

First, having unique user interfaces for all types of devices is impossible. Depending on screen size, bandwidth, and interaction model, customized user interfaces are required. However, HTML is a proper technology for implementing user interface for most types of devices, since a decent Web browser has already been equipped in most mobile devices. From most of the developers' point of view, implementing user interfaces in HTML is also less complicated if compared with other technologies.

Second, user interface processes is a good means to separate presentation from business logic and to offer flexibility to interface customization. Implementing a new type of user interfaces is just a task of implementing another process. Different types of user interfaces can be developed by different people in parallel and deployed at the same time. By this way, user interfaces could also be seen as a service provided by the process. The user interface service is a first class service in a Web Service enabled environment.

Third, BPEL is good technology for implementing Web service based, long-running and process-oriented systems. Especially, the deployment mechanism supported by BPEL implementations offers a very flexible way to deploy various customized user interfaces.

Fourth, since lacking SOAP support from mobile devices or considering performance issues, conversion between SOAP and other protocol is still needed. Communication adapters are crucial parts in the whole system.

Fifth, a controller is not necessary to be implemented as a separate component. Combining a

controller and view generation into a single process is usually more practical.

5 FUTURE RESEARCH

A) User interface as a service. As demonstrated in the case study, user interfaces can be implemented and published as a service. So far, we have only presented the concept of user interface services, but it will be more valuable to investigate and evaluate this concept in the context of the ecosystem of Web Service and Service Oriented Architecture.

B) Applying new user interface technologies of mobile devices. Nowadays, many new user interface technologies are emerging and maturing. For example, AJAX, XForms, SVG, and Flash Lite could be used in the future. However, from the developers' perspective, applying those new technologies also brings new challenges. One of the challenges we are particularly interested in, regarding mobile business processes, is how flexible an end-to-end architecture should be in order to embrace those technologies? The architecture should be able to support both centralized and distributed user interaction logic and be able to help designers and developers to make decisions on how to distribute the interaction logic according to certain requirements.

C) Methodology and tool support for modeling process-oriented system. A business process system is a process-oriented system, which consists of processes and collaboration among processes. There are already methodologies and tools in existence for modeling such systems, like Pi calculus (Smith, 2003). However, we still need to investigate whether there are new requirements or what kind of requirements are for the process modeling technologies, when user interfaces can also be implemented in processes or even when the processes can be deployed on mobile devices.

6 CONCLUSIONS

In this paper, we have focused on how to provide user interfaces agility for developing heterogeneous device enabled business processes by identifying key requirements and proposing five guidelines for designing such business process systems. The highlights from the presented research are the architectural guidelines for developing systems of heterogeneous device enabled business processes and the introduction of concepts of user interface process and user interface service.

REFERENCES

Andrews T., Curbera F., Dholakia H., Goland Y., Klein J., Leymann F., Liu K., Roller D., Smith D., Thatte S., Trickovic I. and Weerawarana S. Business Process Execution Language for Web Services. 5 May 2003. http://dev2dev.bea.com/technologies/webservices/BPEL4WS.jsp

Chakraborty, D. and Hui Lei. 2004. Pervasive Enablement of Business Processes. Pervasive Computing and Communications, 2004. PerCom 2004. *Proceedings of the Second IEEE Annual Conference on 2004* Page(s):87 – 97

Van Gurp Jilles, Karhinen Anssi, Bosch Jan. 2006. Mobile Service Oriented Architectures (MOSOA). DAIS 2006, LNCS 4025, pp. 1 – 15, 2006.

Liang, Z. L. and Wong, R. K. 2005. A lightweight mobile platform for Business Services Networks. *Proceedings of the IEEE Eee05 international Workshop on Business Services Networks (Hong Kong, March 29 - 29, 2005)*. ACM International Conference Proceeding Series, vol. 87. IEEE Press, Piscataway, NJ, 12-12.

Lican Huang, David W. Walker, Omer F. Rana and Yan Huang. 2005. Dynamic Invocation, Optimization and Interoperation of Service-oriented Workflow. Work-in-Progress Section, the 5th International Symposium on Cluster Computing and Grid Computing, Cardiff, Wales, U.K. May 2005

Velez, I.P. and Velez, B. 2005. Lynx: An Open Email Extension for Workflow Systems Based on Web Services and its Application to Digital Government. Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on19-25 Feb. 2006 Page(s):160 - 160

Smith, H. and Fingar, P. 2003. Workflow is just a pi process. http://www.bpm3.com/picalculus