Creation of value added services in NGN with BPEL

T.Eichelmann[1, 2], W.Fuhrmann[3], U.Trick[1], B.Ghita[2]

[1] Research Group for Telecommunication Networks, University of Applied Sciences Frankfurt/M., Frankfurt/M., Germany
[2] Network Research Group, University of Plymouth, Plymouth, United Kingdom
[3] University of Applied Sciences Darmstadt, Darmstadt, Germany
e-mail : eichelmann@e-technik.org

Abstract

The telecommunication industry has recognized the potential of value added services. To be competitive, the companies have to react fast on market changes and on emerging trends. They constantly have to offer new services which are requested by the customers. Also smaller companies who maybe have nested in a niche have to adjust and extend there services. However for the implementation of value added services experts are needed and the implementation of a new service is very time consuming. Existing tools either are coarse-grained or do not offer modern modes of communication such as videoconferencing or web services required by value added services. In this paper a novel approach is presented, showing how services can be designed with the business process execution language (BPEL) and implemented with a service creation environment (SCE) within a short period of time, so that only BPEL-knowledge is required to build a new service. With this approach the service providers are able to offer custom-made services considering their customers requirements and conceivabilities.

Keywords

value added services, service creation, BPEL, JAIN SLEE, NGN

1. Introduction

In the past, telephony comprised simple audio connection between two participants. With the new emerging possibilities like videoconferencing, instant messaging, presence or web services value added services has broadening their scope. Service developers need new tools and development frameworks which can cope with the new requirements.

In the last years some frameworks and tools were developed, supporting the creation of value added services. However the problem of those tools and frameworks is that for the creation of value added services, specialists with extended knowledge are needed and the development process takes a lot of time.

For this reason description languages and graphical service creation environments were developed. The disadvantage of these description languages are their limited possibilities. Often it is only possible to combine prefabricated services into new more powerful services, resulting in a strong coarseness. Furthermore, today's services also require interfaces to other resources such as web services and databases. Also audio conferencing and video conferencing should be available.

This paper proposes a service creation environment that makes the following scenario possible. A customer has an idea for a service and formulates his service idea in textual form. His service provider analyses the service and designs it with the help of BPEL (described in chapter 2.1). Therefore the provider uses the building blocks (described in chapter 3) that are represented by the partner links within BPEL. The resulting BPEL process is not necessarily executable on a BPEL-engine. The service creation environment only uses the resulting xml-files from the BPEL process and transforms the BPEL files to java code that is deployable on an application server (described in chapter 2.2).

2. Used Technologies

2.1. BPEL

The business process execution language is a XML-based language to specify business processes. The activities of the business processes are implemented as web services. BPEL was defined by Microsoft, IBM and BEA in 2002 as BPEL4WS 1.0 (BEA *et al.*, 2002). Version 2.0 was defined by OASIS as WS-BPEL 2.0 in 2004 (OASIS, 2004).

BPEL processes can invoke other web services and they can be invoked from other web services (Figure 1). These services are called partner. A partner communicates with the BPEL process through the web service. This web service interface is defined in the web service description language (WSDL). The simple object access protocol (SOAP) is used as transport protocol between the web services.

There are two ways BPEL processes can be described. One way is to describe an executable business process, which can be run on a BPEL process engine. The other way is the description of an abstract business process.

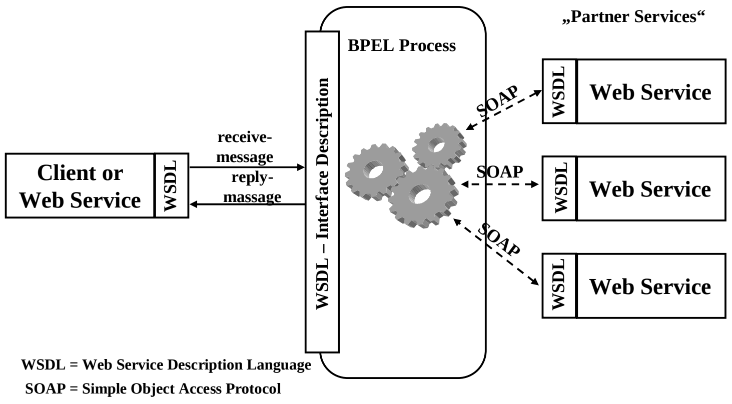

Figure 1: BPEL WSDL Interface

Abstract business processes have the objective to describe only the external visible aspects of the BPEL process. To interact with other web services the BPEL process has to describe the external behaviour which is relevant for the partners. In this approach abstract business processes are used to describe the communication building blocks as partners of the main BPEL process. These building blocks are implemented in java and are not required to be implemented as executable web services in BPEL.

2.2. JAIN SLEE

JSLEE or JAIN SLEE (Sun and Open Cloud, 2008) is the Java standard for SLEE (Service Logic Execution Environment). JAIN means Java APIs for intelligent networks. JSLEE is a high throughput, low latency event processing application environment for communication services. It allows the implementation of scalable high available communication services. Figure 2 shows an overview of the JAIN SLEE application server. The access to network resources is offered by resource adaptors (RA). The services are represented as service building blocks (SBBs). Invocations are transmitted asynchronously via events.

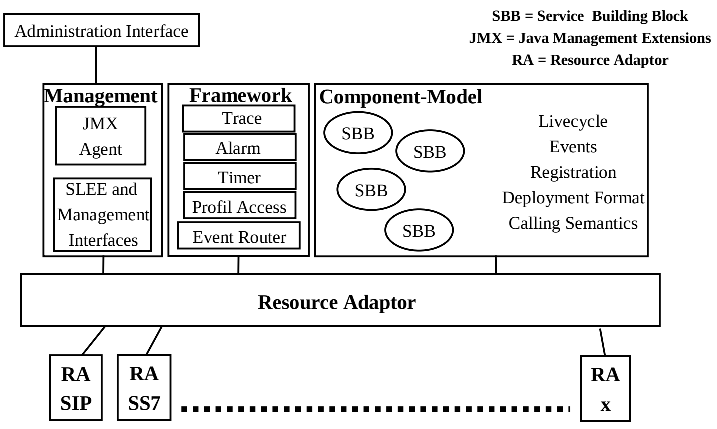

Figure 2: JSLEE Overview

3. Communication Building Blocks

This paper proposes communication building blocks which are made available for the service designer in BPEL as partners. These partners are connected with the process via partner links. Normally external resources such as web services are bound to these partner links.

The partners are communicating with the BPEL process over the WSDL interface. To allow for communication between the BPEL process and the building blocks a web service interface for the building blocks has to be defined. If the functionality of the building blocks increases, the corresponding WSDL interfaces must also be extended.

The building blocks categorize the functionalities provided by the Jain SLEE environment and its resource adapters. The goal was to find a small number of re-usable communication building blocks which cover all necessary functionalities. The functionalities are subdivided into eight communication building blocks: Audio, video, text, files, conference, data input, data output and data trigger. With the combination of some or all of these eight building blocks every value added service can be built.

4. The Service Creation Environment

4.1. Overview

To create custom-made value added services the customer formulates a workflow of the service idea and describes all requirements and conceivabilities for the value added service. His service provider analyses the service and designs a custom-made service with the help of BPEL.

The communication building blocks in BPEL are represented as partners. From these partners the designer can use the necessary methods by invoking them on the communication building blocks. The finished BPEL process is handed over to the service creation engine. There a parser examines the files and builds the service from it.

The service consists of the java classes that represent the JSLEE service building blocks and the corresponding descriptor-files. These descriptor-files are xml-files required for deploying the services and offering configuration options. Java classes, descriptor-files and other dependencies as for example libraries are all copied together into the same workspace. There a compiler creates the class files of the service. After compiling the service creation in completed and the service can be deployed on a JSLEE application server. The service creation process is shown in Figure 3.

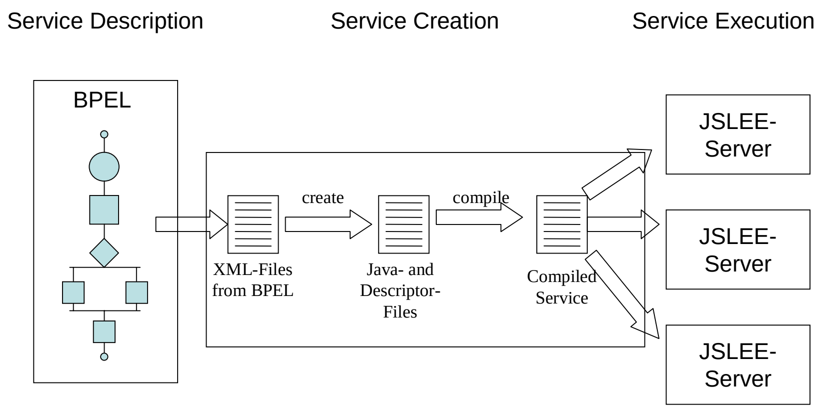

Figure 3: Overview of service creation

4.2. Service description in BPEL

The service is described in BPEL. The communication building blocks are represented in BPEL as partners, shown in Figure 4. The links between the BPEL process and its partners are described by partner links.

The partner links are described in WSDL files. Therefore, for every building block a WSDL description is needed to define the available methods and attributes, so that they become available in BPEL. The building blocks are abstract BPEL processes. The implementation of an executable BPEL process is not necessary, only the message exchange between partners is required to be defined.

Information from the building blocks arrives as events. Information for the building blocks is transported via method calls to the communication building blocks. Within the compiled service a building block calls the corresponding Resource Adaptor which sends out the protocol-specific information. The resulting BPEL process is not executable on a BPEL process engine. The BPEL process design tool produces as output the *.wsdl, *.xsd, and *.xml files. These files serve as input for the service creation.

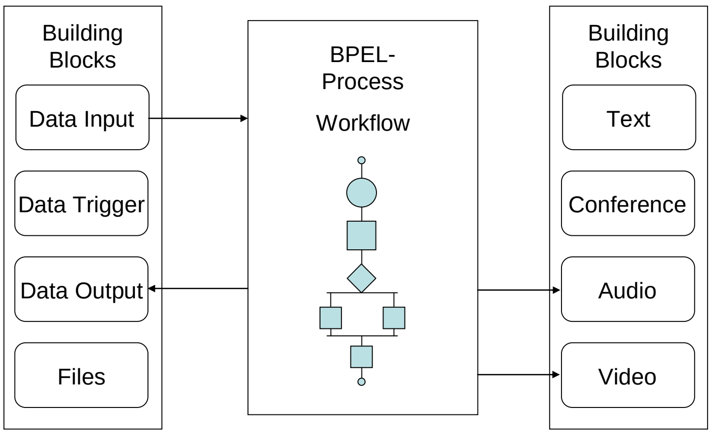

Figure 4: BPEL and the Communication Building Blocks.

4.3. Service Creation

The service creation will transform the BPEL process to java source code. The resulting files from the BPEL process design tool are serving as input. A parser will parse the BPEL files step by step. For every method called on a communication building block, the parser will add predefined code-snippets to the java code of the JSLEE SBBs and xml elements to the descriptor-files.

The descriptor-files describe the elements of the service, such as the SBBs, the events, and the RAs. The communication building blocks used in BPEL are defined as Java code for JSLEE. The java representation of the communication building blocks comprises the resource adaptor calls. Every building block is mapped to one or more resource adaptors. The BPEL activities and the service workflow are also mapped to java code snippets. For every element in BPEL a predefined java equivalent is required. These java snippets are structured in XML-Format. For every BPEL element a list of xml-java code snippets exists which is dedicated to these element.

When the parser reads the BPEL process, for every element that was found, the associated xml-java snippets are added to the source code. For every new resource adaptor to be added to the service creation these xml-java snippets have to be defined. Therefore the java code of the resource adaptor has to be transformed into the corresponding xml format. The new methods and attributes which are needed to access the RA from the building blocks must be added to the corresponding BPEL partner link.

The service creation will create a complete workspace with the needed java code, descriptor-files, build-files, and libraries. After the creation of the java files and the workspace the java compiler creates the service. Now the service can be deployed on a JSLEE application server. An overview of the service creation is shown in Figure 5.

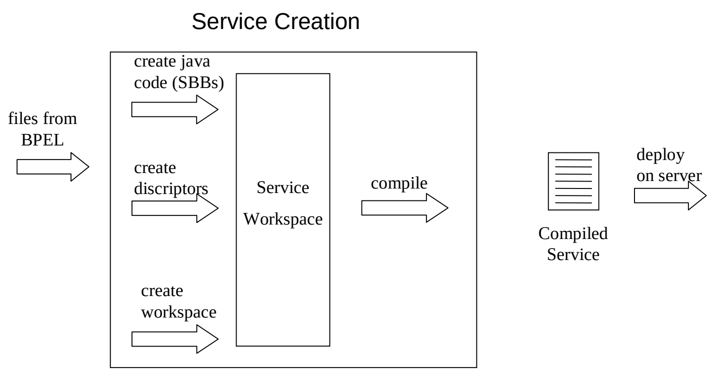

Figure 5: Service Creation

5. Comparison with other Approaches

5.1. Call Processing Language (CPL), Language for End System Services (LESS), Session Processing Language (SPL) and VisuCom

CPL (Rosenberg *et al.*, 1999) is an xml-based language. It mimics the structure of IN service creation. Users can write a script or use a graphical design tool to develop a service. To make the service available, the user has to send it to his service provider.

LESS (Wu and Schulzrinne, 2003) is, like CPL, an xml-based language. It is easy to understand also for non-programmers and offers safety, simplicity, and extensibility.

It contains commands and events that provide direct interaction and control of media applications to users, media applications and other end system applications.

SPL (Burgy *et al.*, 2006) is a domain-specific language which allows for the development of robust telephony services and offers an abstraction over the underlying protocols and software layers.

VisuCom (Latry *et al.*, 2007) is a graphical telephony service creation and execution environment over SPL. It enables non-programmers to define services. It offers intuitive visual constructs and menus that permit users a quick development of telephony services.

However the scope of these approaches is limited to end-user services and the scripting languages are mostly coarse-grained. The approach defined in this paper allows a finer granularity and a wider range of communication modes. New resources can be added through JAIN SLEE by implementing new resource adapters. To enable access to the new resources, java code snippets have to be defined and the methods to control the new resources have to be added to the communication building blocks.

5.2. DiaGen

DiaGen introduces a declarative language over Java. Out of the declarative language, DiaGen (Jouve *et al.*, 2008) generate a framework for the java programming language. This framework provides service discovery and high-level communication mechanisms. The framework generates the class skeletons of service classes. The programmer must fill in the service code.

However with the approach described in this paper a programmer has to write the java code only once. Then the resulting code snippets are mapped to the correspondent building blocks. New resources can be added through new resource adapters. These resource adapters can be developed by the service provider or acquired from the vendor of the JSLEE application server.

6. Conclusion and the Future

In this paper a new service creation environment was proposed. This work should overcome problems shown by other approaches. The goal of this SCE is a rapid and easy creation of customer-made value added services. The creation of new services can be achieved by service designer who understand the business process execution language.

The communication building blocks provide a simple, comfortable possibility for the designer to create the BPEL process. The SCE transforms the BPEL process to java code and builds the class-files. The JSLEE application server with its resource adapters offers defined interfaces which can mapped to the building blocks in BPEL.

The combination of JSLEE, BPEL, and the service creation engine allow a rapid development of value added services. The next step is the definition of building block methods with the corresponding java code snippets. A XML-structure has to be defined to structure the code snippets for code generation.

7. Annotation

The research project providing the basis for this publication was partially funded by the Federal Ministry of Education and Research (BMBF) of the Federal Republic of Germany under grant number 1704B07. The authors of this publication are in charge of its content.

8. Acknowledgment

I thank Armin Lehmann, Björn Ricks, Rolf Lasch and Patrick Wacht for the discussion and criticism of this work.

9. References

BEA Systems, IBM, Microsoft, Specification V1.0 (2002), "Business Process Execution Language for Web Services", IBM developerWorks

Burgy L., Consel C., Latry F., Lawall J., Palix N., Réveillère L. (2006), "Language Technology for Internet-Telephony Service Creation", *IEEE International Conference on Communications 2006*, Instanbul, ISBN: 1-4244-0355-3

Jouve W., Palix N., Consel C., Kadionik P. (2008), "A SIP-based Programming Framework for Advanced Telephony Applications", *Principles, Systems and Applications of IP Telecommunications Services and Security for Next Generation Networks,* IPTCOM 2008 Heidelberg, Germany 2008

Latry F., Mercadal J., Consel C. (2007), "Staging Telephony Service Creation: A Language Approach", *In Principles, Systems and Applications of IP Telecommunications*, IPTComm, New-York, NY, USA, July 2007, ACM Press

OASIS (2004), OASIS Standard, "Web Services Business Process Execution Language Version 2.0", OASIS

Rosenberg J., Lennox J., Schulzrinne H (1999), "Programming Internet Telephony Services", *IEEE Internet Computing Magazine*, March 1999

Sun Microsystems, Open Cloud (2008), JSR-000240 Specification, Final Release, „JAIN SLEE (JSLEE) 1.1", Sun

Wu X., Schulzrinne H. (2003), "Programmable end system services using SIP", *IEEE International Conference on Communications*, May, 2003, 2nd New York Metro Area Networking Workshop, September 2002