TeamCom: A Service Creation Platform for Next Generation Networks

A. Lehmann, T. Eichelmann, U. Trick
Research Group for Telecommunication Networks
University of Applied Sciences
Frankfurt/Main, Germany
e-mail: {lehmann, eichelmann, trick}@e-technik.org

R. Lasch, B. Ricks, R. Tönjes
Research Group for Mobile Communications
University of Applied Sciences
Osnabrück, Germany
e-mail: {r.lasch, b.ricks, r.toenjes}@fh-osnabrueck.de

Abstract— The development of value added services is currently still very time and cost consuming. The need for specific user generated and in particular business-to-business services demands for efficient service development methods. This paper presents a service creation environment that supports the application developer to compose a service based on reusable components and to describe the business process through a control logic. Forthe service description a language that has been optimized for business prozesses is suggested: the Business Process Execution Language (BPEL). However, BPEL has not been developed for control of specific, in particular real time, communication services in heterogeneous networks.Therefore the paper presents a parser translating the business process description into Java code and supporting the deployment of the service in a service execution environment based on JAIN SLEE. The provided elementary communication Service Components hide the underlying heterogeneous communication networks. Thereby the developer does not need any detailed knowledge of communication protocols and is able to focus on the application logic instead. This leads to new opportunities for rapid and efficient service creation using a new Service Creation Environment (SCE) with higher level of abstraction and automated service generation.

Keywords-component; SCE (Service Creation Environment; NGN (Next Generation Networks); Service Components; JAIN SLEE

I. INTRODUCTION

The provision of customer specific communication processes is currently still very time and cost consuming. As a consequence the penetration of specific multimedia services is low. This holds for business-to-business (B2B) services as well as for individual services, which are even in the Web 2.0 area limited to simple services. Hence many capabilities of today's multimedia networks remain often unexploited.

B2B services offer a high potential to allow for the acceleration of processes and workflows within and between organizations. Unfortunately, the development of mobile B2B services requires still a lot of detailed knowledge about mobile communication systems and their protocols. Moreover the application developers need a deep understanding of the embedded business processes. The wide range of the necessary knowledge hinders the growth of mobile B2B services and fosters proprietary solutions. Therefore the TeamCom project [1] aims at enabling fast, easy and cost efficient provisioning of value added services, in particular B2B services, by creating a new high level Service Creation Environment (SCE). Elementary communication Service Components are derived from the requirements for telecommunication services. To support the service developer these communication components should be provided and integrated into a generic Service Creation Environment for mobile B2B services in heterogeneous networks.

For the orchestration, i.e. composition, of services various SCE have been suggested recently in the literature. As lessons learnt the European SPICE [2] and OPUCE [3] projects point out the need for a clear separation between professional and end-user service creation. The LOMS [4], the MAMS [5] project and the Open Source Initiative SPAGIC [6] have proven the advantages of a graphical service creation workbench, developed in particular for small and medium sized enterprises, to ease B2B service creation. To cope with evolving service requirements and the heterogeneity of the underlying communication and computing infrastructure, respectively, the SeCSE [7] and the PLASTIC [8] project suggested self-adapting service oriented applications.

This paper takes a systematic approach for service creation by reusing communication Service Components and suggests building the service creation environment on top of the future control layer of next generation networks, the IP Multimedia Subsystem (IMS) [9], which is based on the Session Initiation Protocol (SIP) [10]. For the first time, IMS offers the opportunity to provide services uniformly over heterogeneous networks (cellular networks, WLAN and fixed networks) and outside of the user's home network. It provides an integrative support for mobility, quality of service, security and service dependent accounting. In contrast to this centralized approach peer to peer (P2P) communication, e.g. Skype and P2P-SIP [11], could provide a cost efficient and scalable alternative.

To speed up the development of services, in particular of B2B services, this paper proposes a fourfold approach:

- Abstraction of heterogeneous network interfaces
- Definition of reusable communication Service Components
- Development of a Service Creation Environment combined with automated service generation employing reusable components
- Use of partial services by cooperating application servers

The reminder of this paper is structured as follows: The next chapter presents the overall architecture of the system. Chapter III describes the generic Service Creation Environment and chapter IV discusses the elements for supporting the service life cycle as a whole. Chapter V shows an example of service creation and deployment evaluating the approach and chapter VI finally concludes the paper.

II. Architecture

The TeamCom Project proposes an integrated architecture for service creation, deployment and execution (see Fig. 1). This architecture can be divided into four layers: (1) The Service Creation Environment (SCE), which includes the design and composition of new services, (2) the Service Deployment (SD), (3) the Service Execution Engine (SEE) containing one or more Application Servers (AS) based on JAIN SLEE (JAIN Service Logic Execution Environment) and (4) the Service Transport Layer (STL). The Service Transport Layer abstracts different protocols in order to enable upper service layers to be independent of a specific communication protocol. Therefore it supports several communication networks, e.g. IP Multimedia Subsystem or Peer-to-Peer SIP. The Service Creation Environment contains a GUI, permitting a developer to orchestrate Service Components and to integrate external services easily without having to develop low-level software code. The Service Execution Environment establishes a layer where created Services Components are deployed and activated.

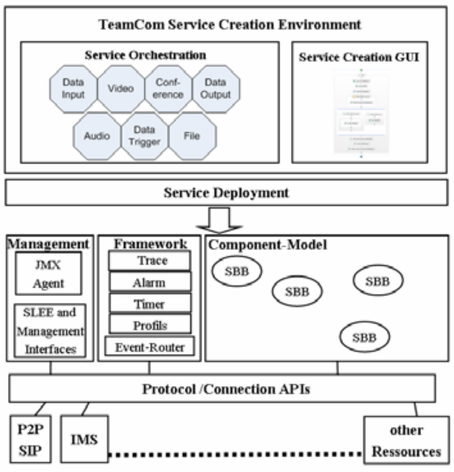

Figure 1. TeamCom Architecture

In order to support Next Generation Networks that are based on IP, different software architectures for the Service Execution Environment have been evaluated. The following requirements for a Service Execution Environment have been defined: independence from the operating system, service orchestration based on components, comfortable component deployment, support for SIP, extensibility for other protocols and possibility for co-operation between numerous application servers. To fulfil these requirements the architecture makes use of the JAIN SLEE standard [12].

The JAIN SLEE standard defines a component, event and transaction based architecture. The architecture is written in the Java language and standardised by a Java Community Process. It is part of the JAIN (Java API for Integrated Networks) Initiative consisting of several telecommunication companies. JAIN SLEE is designed for ensuring low latency and providing high throughput to accomplish the requirements for communication services. It uses a distributed component model similar to Enterprise Java Beans (EJB) [13]. Therefore it is often referred as "EJB for Communications".

In the standard so called Resource Adaptors (RA) are defined abstracting the underlying infrastructure. These Resource Adaptors provide a common Java API which hides the communication protocol underneath. In detail when a communication protocol message is received the corresponding RA translates this message into a Java event class. Afterwards the event and an activity class both together are passed to the JAIN SLEE event router. A JAIN SLEE activity represents a session of a communication protocol e.g. a SIP dialog. The event router utilizes these objects to look up the services which requested to receive the specific event. Accordingly the service itself is able to react on the event and to create an answer by using defined Java Interfaces. The answer is translated to a communication protocol response by the RA. The JAIN SLEE standard pre-defines (among others) the interface for a SIP Resource Adaptor. Additional Resource Adaptors can be added manually. In TeamCom several Resource Adaptors for email communication (Mail RA), for evolved Web Service access (Web Service RA), for security mechanisms based on TLS (Transport Layer Security) (TLS RA) and IMS extensions for SIP (IMS RA) have been developed. Hence a service based on JAIN SLEE is able to support heterogeneous networks.

The service itself is composed of one or more Service Building Blocks (SBB). These SBBs contain the application/service execution logic and are deployed on a JAIN SLEE Application Server such as Mobicents [14]. The SBB component model includes a lifecycle, registration and security management. In addition, SBBs are able to access timer, trace, alarm and profile facilities which are also provided by a JAIN SLEE server.

III. Service Creation Environment

As depicted in the last chapter the Service Creation Environment includes service orchestration on the basis of reusable Service Components and existing services. In addition to the components a service logic, similar to “if then” statements, is required for describing the workflow properly. The following two sections explain both.

A. Reusable Service Components

Several elementary Service Components are derived from the requirements for telecommunication services. By combining these elementary components it should be possible to describe and generate other valuable services. These elementary Service Components comprise audio, video, text, file, conference, data input, data output and data trigger components. This chapter discusses the abstract Service Components in detail.

Audio: The Audio Component handles all kind of audio communication including the establishment of a call, answering calls, manipulation of audio streams (e.g. mixing, transcoding) and sending and receiving DTMF tones.

Video: The Video Component is responsible for playing and recording of video streams. It enables to create and close video calls and to combine different video signals for merging a new video stream.

Text: This component exchanges messages between two partners and has the capabilities of handling strings, e.g. search for a specific word in a text, replace alphabetic characters or change the encoding of a text.

File: The File Component handles creation, deletion, sending and receiving of binary files. Another task of File is to write and read any kind of data from and to any position in a file. Finally this component is able to rename files or directories.

Data Input: All kind of data queries are processed by the Data Input component. This includes database queries as well as reading data from a sensor.

Data Output: The counterpart of Data Input is Data Output being concerned with writing data to a destination e.g. to a database or to control an actuator.

Conference: This is a special kind of communication component because it re-uses internal functions of the previously described components, e.g. audio stream mixing. On top of this, the Conference Component provides functionalities for creating and deleting conference “rooms”, adding and removing users to/from a room.

Data Trigger: The Data Trigger is closely related to an event generator. If a specific data trespasses a value an event is triggered. This data can be a sensor value, a timestamp or periodical dates.

B. Service Logic

Business workflows within and in between enterprises usually follow defined processes, the so called business processes. This resulted in the definition of the BPEL (Business Process Execution Language) standard [15], an XML [16] based process description language, building completely on web services.

Most often BPEL is used to orchestrate web services but the language is protocol independent. It is possible to create so called bindings which could specify the usage of BPEL for other protocols than SOAP (Simple Object Access Protocol) [17]. In BPEL it is possible to define synchronous and asynchronous processes. Thus it fits very well to be used in combination with JAIN SLEE. For a structured process different BPEL tags are defined, e.g. sequence, if, while. Moreover forked processes can be created, allowing for execution in parallel and synchronisation afterwards. As depicted in Fig. 2 the BPEL process has the ability to ‘invoke’ other services asynchronously after having received a request from a client. The ‘reply’ BPEL element is used to send an answer to the client.

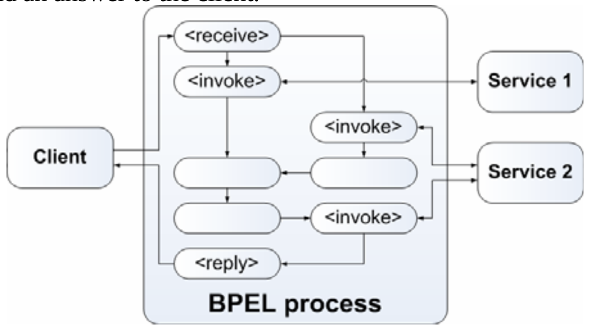

Figure 2. BPEL Process Example

C. Service Creation

To orchestrate value added services easily a graphically interface is provided by the TeamCom Service Creation Environment. It is possible to drag and drop Service Components to the workspace where the BPEL process is designed. Additionally BPEL elements can be dragged into the workspace and be connected to the Service Components. TeamCom employs BPEL to compose and control the sequence of the Service Components. All defined Service Components are provided as partner links in BPEL. Partner links define the communication partners and their roles in a process. Our Service Creation Environment only relies on a subset of all BPEL commands and doesn’t require a BPEL engine. So it could be possible to choose another language for the TeamCom SCE to describe a service then BPEL because it is only used to define the service logic. Literature has shown that other service description languages are often to protocol specific and/or not extendable (e.g. Call Processing Language) [18] [19] or they were withdrawn (e.g. Service Control Markup Language) [20].

After configuring all elements and components it is possible to create a service which can be deployed on an Application Server.

The Service Components are described in the Web Services Description Language (WSDL) [21] which is also based on XML. Therefore all components provide an interface independent of a communication protocol. In WSDL the methods of the Service Components are defined e.g. the ‘onCall’ method of the Audio Component. It is possible to support different implementations of Service Components, depending on the desired communication system. If a service should be deployed in an IMS network it is necessary to implement an Audio Component which is capable of the IMS SIP extensions. Thus every implementation has to create a WSDL binding, which describes the mapping between the abstract WSDL operation and the implementation.

The interfaces of a BPEL process itself are also described by WSDL. Therefore it is possible to include an external BPEL process in a newly created BPEL process. As a result small, simple and more generic services could be created which are reusable in other services. Both the integration of external services and Service Components is carried out in the same way.

IV. Service Life Cycle

The ability to manage the lifecycle of B2B services is fundamental for achieving success within mobile service platforms. The service life cycle can be subdivided into a succession of operations that include service creation, deployment and execution. The following section introduces the selected Service Execution Environment before deriving the associated creation and deployment process.

A. *Service Execution Engine*

The B2B services require a Service Execution Engine that can conduct the business processes on top of the telecommunications infrastructure. The question rises how these two worlds can be combined.

In telecommunications a plethora of protocols and standards have to be supported (here in particular SIP and IMS). In such a heterogeneous environment, the services should be decoupled from the various underlying communication networks. The JAIN SLEE standard provides resource adaptors that abstract from the various underlying networks.

Complementary BPEL engines allow executing BPEL process instances. For this the BPEL processes are deployed on the BPEL engine using engine specific deployment information.

Four different approaches were analysed for accessing the BPEL information within the JAIN SLEE, which are depicted in Fig. 3:

1. Master SBB reads an XML file (e.g. BPEL).
2. Access of BPEL engine over resource adaptor.
3. Access of BPEL engine over J2EE connector.
4. Translation of BPEL into JAIN SLEE logic

The direct use of XML files is quite static. Also the files must be parsed during the execution time which means lower performance. In contrast alternative two and three would keep the BPEL engine allowing to run the workflows as BPEL processes. In alternative two BPEL would be treated like a protocol and invoke an event over the resource adaptor [22]-[24]. Alternative three employs the J2EE Connector Architecture (JCA) [25] providing via EJB (Enterprise Java Beans) an interface to the Java based BPEL engine operating in a J2EE environment. However, both alternatives with external BPEL engines limit the real time capabilities of the service. Also the usage of a BPEL engine will not fit to a P2P model, because the peers can not benefit by the BPEL engine.

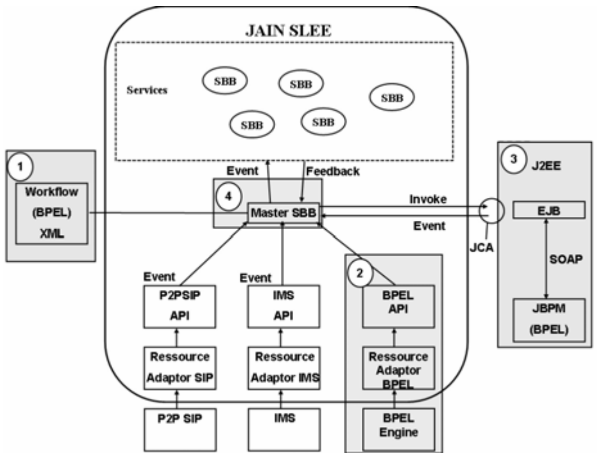

Figure 3. Integration of BPEL and JAIN SLEE

Therefore the fourth alternative was chosen, that translates the resulting BPEL files into a JAIN SLEE service. This means that the service which will be created is designed with a graphical editor based on BPEL elements. Within this solution no BPEL engine is used during the execution of the service, which will fasten up the execution time and also support P2P networks. Please note that in this case the Service Components can be integrated with the control information in only one (or a few) SBB, providing the service. This approach is detailed in the next section.

The SEE may provide different services, i. e. specific business processes, in parallel. Moreover, the B2B service may use external service parts that are distributed over different Application Servers.

B. *Service Creation and Deployment*

Service creation and deployment can be realised in five steps: writing a non-technical description of the service, converting this description to a service description language, analysing the description language, generating Service Building Blocks from the description language and deploying the service. First a business description has to be verbalised. The description could be e.g. a text, depending on the process and the involved people. Afterwards a service developer is able to create a BPEL based description graphically via the service creation Graphical User Interface (GUI). In this step the service developer utilizes the TeamCom Service Components which are included in BPEL as partner links and configures their input variables e.g. video sender address. After finishing the BPEL process description the generation of the service will start (see Fig. 4).

The service designed by the service developer shall not be executed on a BPEL process engine. Instead the service shall run on a JAIN SLEE application server. So the BPEL process has to be converted in a form to be executable as a JAIN SLEE service. The code generator analyses the BPEL process and parses the workflow step by step. The result from each individual step is saved in a XML service document that could be described as an intermediate language for a JAIN SLEE service. Finally the goal of the code generator is the creation of the Java classes and the necessary descriptor files needed for a JAIN SLEE service.

While the code generator parses the BPEL process, it analyses the BPEL activities. Pending on the BPEL activity the code generator adds pre-defined XML fragments to the XML service document. Process activities which initiate events for the partners or waiting for events from them must be examined to figure out which Service Component is affected by this event. For each method which can be invoked on a partner a pre-defined XML fragment in a XML fragment pool exists. If the Service Component is identified, an appropriate XML fragment which represents the used method from the BPEL process can be chosen from the XML pool and inserted into the XML service document. Other workflow activities perhaps need variables or data structures which are defined in BPEL. These structures are represented in XML schemata and have to be transformed into Java code also.

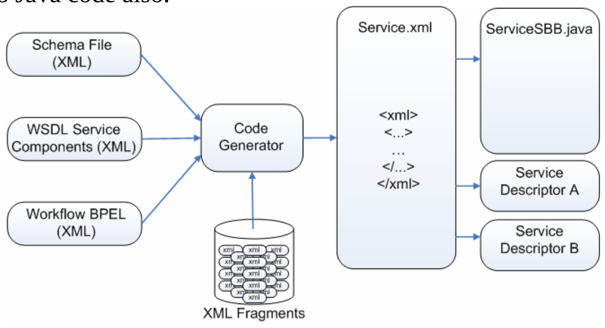

Figure 4. Code Generation

In Table I some examples for BPEL activities and their correspondent part in Java are shown. Now the XML service document contains all information necessary to generate the Java code, the service descriptor files and the build file.

The code generator will not create an intermediate language in future versions which will result in better performance. Instead of the XML fragments Java templates will be used and the necessary Java classes, a build file for the deployment and the descriptor files will be generated immediately.

TABLE I. COUNTERPARTS OF BPEL ACTIVITIES AND JAVA

BPEL Activity	Java
if	if()
while	while()
forEach	for(;;)
repeatUntil	do{}while()
wait	SLEE timer
sequence	sequential flow
flow	parallel flow
invoke	bidirectional method
receive	listener, getter method
reply	setter method
assign	operations e.g. String operations

.

C. *Evaluation of Service Creation*

The described service creation process has to be evaluated to validate the code generator. Therefore different scenarios were estimated. Those scenarios cover a maximised set of telecommunication services. The following enumeration is a subset of services to be evaluated:

- Session based services (e.g. active, passive, parallel or sequential sessions)
- Conferences (e.g. audio and/or video)
- Call-control (e.g. forwarding, blocking, forking)

These scenarios will be designed with the BPEL editor and generated as described before. The resulting Java source code has to be verified. So the evaluation will be an empirical process.

V. EXAMPLE

To demonstrate the TeamCom Architecture an exemplary and simple wake-up service has been developed. The Text Component is used to receive messages from a SIP client. These messages contain the date when the user will be woken and a text phrase the user wanted to receive. The application processes the incoming messages and passes the date to the Data Trigger Component. The timer will trigger the service at the reached date and finally our service creates a wake-up SIP MESSAGE which will be sent back to the user.

Fig. 5 depicts the creation of this wakeup service in the TeamCom SCE workbench showing on both sides the Service Components embedded in the service. The Text Component is responsible for receiving and sending messages. The Data Trigger Service Component is used for generating a timer.

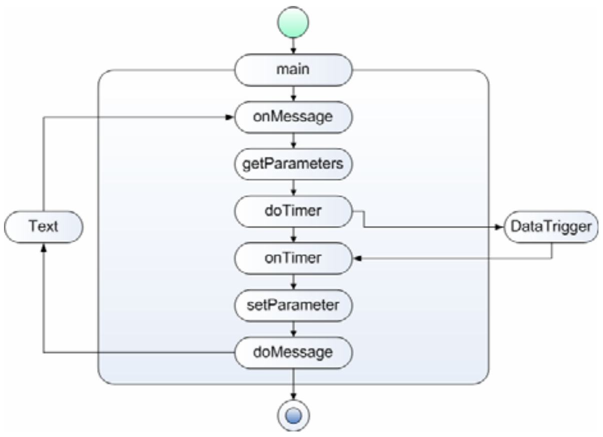

Figure 5. Wake-up Service Example

The main sequence contains the entire description of the service. First the service is waiting for an incoming text message from a client, which is handled by the ‘onMessage’ element of the Text Component. The received text message contains the date and text for the wake-up call. In ‘getParameters’ the BPEL assign element is used to get the necessary data for the timer. Afterwards the invoke element ‘doTimer’ is used to generate a timer via the Data Trigger Component. The element ‘onTimer’ will start the service again after the timer expired. The next element ‘setParameter’ is used to set the text phrase for the following element ‘doMessage’. The reply element ‘doMessage’ is used to generate the wake-up SIP MESSAGE which will be sent back to the originating user.

VI. Conclusion

The development of new services for telecommunication applications and other IT systems is the most important means of innovation for telecommunication service providers. In particular, B2B applications possess a high potential to accelerate processes and workflows within and between organisations. Competition demands for faster innovation cycles to speed up time to market.

The presented SCE empowers small and medium sized enterprises to develop their B2B services in a time and cost efficient way. A graphical user interface supports even inexperienced developers to orchestrate the reusable components for value added B2B services, exploiting the full power of nowadays multimedia networks through predefined resource adaptors. The possibility for automated creation of communication services without a detailed knowledge about the used protocols and networks is given. The presented example was already created and generated by the usage of our SCE. Also different variations of this exemplary service could be developed within minutes. Service creation could be as easy as current website development and applicable for more people. Vice versa the ease of service creation will foster a better degree of utilisation of the telecom munication’s multimedia networks. In the TeamCom architecture elementary communication Service Components are abstracted for a general use in possible Next Generation Networks. The Service Component interface is extensible for evolving protocols and networks.

Acknowledgment

This work has been supported by the German Federal Ministry of Education and Research (BMBF) under grant 1704A07 and 1704B07 in the TeamCom project.

References

[1] TeamCom project website: http://www.ecs.fh-osnabrueck.de/teamcom.html
[2] O. Droegehorn et al.: “Professional and End-User driven Service Creation in the SPICE platform”, IEEE WOWMOM 2008, New Port Beach, California, USA, 23-25 June 2008.
[3] J. C. Yelmo et al.: “A User-Centric service creation approach for Next Generation Networks”, ITU-T Kaleidoscope Event: Innovations in NGN - Future Network and Services, Geneva, 12-13 May 2008.
[4] J. Keiser, T. Kriengchaiyapruk. "Bringing Creation of Context-Aware Mobile Services to the Masses". In: IEEE SOA Industry Summit (SOAIS 2008), Hawaii, USA, July 2008.
[5] B. Freese, U. Staiger, H. Stein: "Multi-Access Modular-Services Framework - Whitepaper", Deutsche Telekom Laboratories, Berlin, June 2007.
[6] SPAGIC web site: http://spagic.org/ecm/faces/public/guest/home/solutions/spagic
[7] L. Baresi, E. Di Nitto and C. Ghezzi, “Toward Open-World Software: issues and challenges”, IEEE Computer, Volume 39, No. 10: 36-43, October 2006.
[8] M. Autili et al.: "A Development Process for Self-adapting Service Oriented Applications", in proceeding of ICSOC 2007(442-448), Vienna, Austria, September 2007.
[9] TS 23.228: IP Multimedia Subsystem (IMS); Stage 2 (Release 5). 3GPP, June 2006.
[10] J. Rosenberg et al.: RFC 3261 - SIP: Session Initiation Protocol. IETF, June 2002.
[11] Web site of IETF P2PSIP WG: http://www.p2psip.org/
[12] Sun Microsystems, Open Cloud, JSR-000240 Specification, Final Release, "JAIN SLEE (JSLEE) 1.1", SUN, 2008.
[13] Sun Microsystems, Oracle Corporation, JSR-000220 Specification, Final Release, "Enterprise JavaBeans, Version 3.0", SUN, May 2006.
[14] Mobicents Open Source JAIN SLEE Server, http://www.mobicents.org
[15] IBM, Microsoft, Specification V 2.0, "Web Services Business Process Execution Language Version 2". OASIS, April 2007.
[16] W3C, "Extensible Markup Language (XML) 1.0 (Fith Edition)", W3C Recommendation, November 2008.
[17] M. Gudgin, et al.: "SOAP Version 1.2 Part 1. Messaging Framework (Second Edition)". W3C, April 2007.
[18] R.H. Glitho, A. Poulin: "A high level service creation environment for Parlay in a SIP environment". IEEE International Conference on Communications, 2002.
[19] R.H. Glitho et al.: "Creating value added services in Internet telephony: an overview and case study on a high-level service creation environment". IEEE Transaction on Systems, Man, and Cybernetics, Part C, November 2003.

[20] J.-L. Bakker, R. Jain: "Next generation service creation using XML scripting languages". IEEE International Conference on Communications, 2002.
[21] E. Christensen et al.: "Web Services Description Language (WSDL) 1.1". W3C, March 2001.
[22] S. Bessler et al.: "An Orchestrated Execution Environment for Hybrid Services". Kommunikation in Verteilten Systemen, vol.II, page.77-88, April 2007, Springer Berlin Heidelberg.
[23] G. Jie et al.: "A Template-based Orchestration Framework for Hybrid Services". AICT '08, Telecommunications, June 2008.
[24] P. Falcarin, C. Venezia: "Communication Web Services and JAIN-SLEE Integration Challenges". International Journal of Web Services Research, Vol. 5, Issue 4, page. 59-78, 2008.
[25] Sun Microsystems, JSR-000112 Specification, Final Release, "J2EE Connector Architecture 1.5", SUN, November 2003