ServiceCom: A Tool for Service Composition Reuse and Specialization

Bart Orriëns, Jian Yang and Mike P. Papazoglou
Tilburg University, Infolab, PO Box 90153, 5000 LIE, Tilburg Netherlands
{b.orriens, jian, mikep}@uvt.nl

Abstract

Web services are becoming the dominant paradigm for distributed computing and electronic business. This has raised the opportunity for service providers and application developers to create value added services by combining web services. Several web service composition solutions have been proposed, e.g. BPEL4WS. However, these approaches are either not flexible or too complicated as they lack proper support for modularity and reusability. Motivated by the demand of a light-weighted tool of service composition, we developed ***ServiceCom****, a tool for service composition specification, construction and execution. In this paper we discuss how ServiceCom suppors reusable web service composition specification, combination, and execution.*

1. Introduction

Web services are self-contained, internet-enabled applications capable not only of performing business activities on their own, but also possessing the ability to engage other web services in order to complete higher order business transactions. The platform neutral nature of the web services creates the opportunity for building *composite services* by using existing elementary or complex services possibly offered by different enterprises. For example, a `travel plan` service can be developed by combining several elementary services such as `hotel reservation`, `ticket booking`, `car rental`, `sightseeing package`, etc., based on their WSDL descriptions. By *composite service*, we mean a service that uses other services. The services that are used by a composite service are called its *constituent services*. Web service design and composition is a distributed programming activity. It requires software engineering principles and technology support for service specification, reuse, specialization, and extension such as those used, for example, in component based software development. Normally composite services are developed by hard-coding business logic into application programs. the development of business applications would be greatly facilitated if methodologies and tools for supporting the development and delivery of composite services in a coordinated and effectively reusable manner were to be devised. Some preliminary work has been conducted in the area of service composition, mostly in aspects of business process modeling language [2] and composition specification e.g. BPEL4WS [3]. However, these approaches are either not flexible or too complicated as they lack proper support for modularity and reusability. Motivated by the demand of a light-weighted tool of service composition, we developed **ServiceCom**, a tool for service composition specification, construction and execution. In this paper we provide an overview of ServiceCom, and outline its significance and contributions to the research in service composition.

2. Overview

ServiceCom is a Java based tool for modular and reusable web service composition. The tool is based on the concept of *service* ***component*** [7], which facilitates the idea of web service component reuse, specialization and extension. ***Service components*** are a packaging mechanism for developing web-based distributed applications in terms of combining existing (published) web services. Service components have a recursive nature in that they can be composed of published web services while in turn they are also considered to be themselves web services (albeit complex in nature). Once a service component is defined, it can be reused, specialized, and extended.

To support the service component mechanism a **Service Composition Specification Language (SCSL)** has been developed (see [7] for the syntax). In this language *activity*, *binding*, *condition* and *composition type* constructs are used (among others) to define a web service composition. *Activity* constructs represent constituent services through the use of name and description characteristics. To bind activities to particular web service providers *binding* constructs can be attached to activity constructs. These bindings identify an operation on a port of a service in the WSDL interface of the provider. Alternatively, they may provide search criteria to enable the locating of appropriate providers during runtime. *Condition* constructs may be used in case of conditional compositions to govern the control flow within the service

component. To express different forms of activity choreographies SCSL utilizes the *composition type* construct. Supported types include If, IfElse, ParaWithSync, ParaAlt, SeqAlt, SeqNoInt, SeqWithInt and WhileDo patterns.

Based on top of the service component mechanism and SCSL **ServiceCom** propagates a phased approach to service composition spanning composition description, planning, building and invocation. An overview of this phased approach is provided in Figure 1. In the following we briefly describe the purpose of the **Description**, **Planning**, **Building** and **Invocation** phase and discuss how ServiceCom implements each phase. We also explain how reuse, extension and specialization are supported.

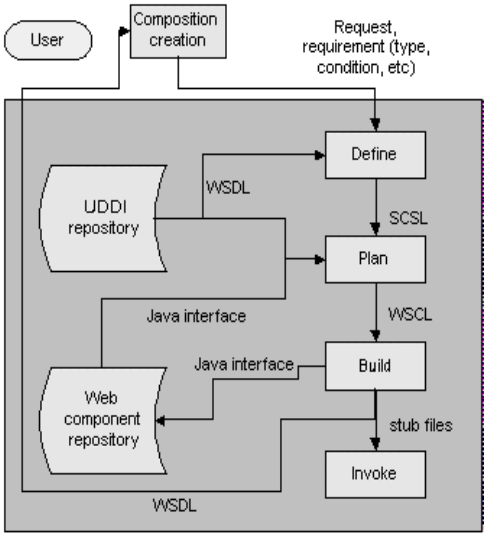

Figure 1. Web service composition phase model

2.1. Description phase

In the *Description* phase users can specify a web service composition. For this purpose ServiceCom offers a **composition designer**, which provides a visual-oriented development environment. First a *composition type* is selected. Subsequently *activity*, *binding* and *condition* constructs may be dragged and dropped on the designer window. Characteristics may be set and edited through the use of popup dialogs, which can be opened by double-clicking a composition construct. Alternatively, constructs may be inserted, edited and removed through the use of menus. A screenshot of the composition designer is provided in Figure 2 in Appendix A.

Reuse, extension and specialization of constructs in the *Description* phase is based upon the use of unique names and namespaces. This combination ensures that a reference to a construct may only refer to a single construct, since a construct name must be unique within a namespace and each composition has a unique namespace. For the reuse of constructs simple references can be used.

Extension and specialization is supported through the utilization of base types. When specifying a base type for a construct the characteristics of this base construct are inherited by the sub-construct. Additional information in the sub-construct not present in the base construct extends the base construct. Similarly, in case of information that is present in both the sub and base construct the characteristics of the base construct are overridden. In ServiceCom both references and base types can be made through the selection of composition constructs from the **web service composition library**. (See Figure 4 in Appendix A for a screenshot)

2.2. Planning phase

The *Planning* phase is concerned with binding the activities in the service composition to concrete services. For this purpose ServiceCom offers a **web service provider library** facility. This library facility contains locally known web service providers, which can aid the user in case he does not know an appropriate provider for an activity in a composition. (i.e. the WSDL interfaces are stored on the local system). Web service providers may be added and removed from the library. Also, a search on the Internet can be performed to locate suitable providers, based on user-specified search criteria. Any located providers may then be added to the library, making them locally available. Furthermore, provider functionality may be tested through a quick invocation. As such, the web service provider library is of great help in the planning phase of the web service process model by enabling the reuse of web service providers. For a screenshot of the library see Figure 3 in Appendix A.

In case a user does not want to search and select a service provider manually ServiceCom is capable of selecting appropriate providers automatically during runtime. For this purpose the tool utilizes the search criteria specified in the binding construct(s).

2.3. Building phase

The *Building* phase is of essence in the web service composition phase model. It crosses the gap between composition specification and actual invocation. In this phase the web service composition specification is used as input to generate the set of Java source and class files required for the invocation of the composition. This set of files comprise of the following:

- Firstly, for each activity in the composition a set of Java source and class files is created for invocation. These files are derived from the WSDL definition of the service provider, specified in the binding construct of the activity. For each service in the

definition a class is defined, as well as for the ports of these services. Operations are represented as Java methods, whereas faults are expressed as exceptions. Additionally, files are generated that translate operation invocations from and to SOAP messages to enable XML-based invocation.

- Secondly, a set of Java source and class files is created for the web service composition. These files enable the invocation of the composition in the specified manner. The exact types of files that are generated depend on the type of the composition. For example, if a composition is of the type i*f*, then its condition constructs are mapped to source and class files. In case of *parallel* compositions source and class files are generated that enable the invocation of an activity in a separate thread of execution.
- Thirdly, a WSDL definition is generated for the web service composition. This definition functions as the public interface of the composition.
- Fourthly, the composition is stored as a SCSL specification to enable further reuse

The above files can all be reused in the composition building process. Firstly ServiceCom will only generate source and class files for a composition if they are not already in the library. For example, suppose a service is involved in multiple compositions, instead of generating new files for each composition only one set files will be generated and it will be reused in all other compositions this service is involved in.

Secondly, generated files can be used in the development of other applications. The mentioned functionality may be extended and specialized in an object-oriented fashion through the use of Java overriding and inheritance mechanisms. However, these two types of reuse are in our opinion best facilitated in professional programming environments and are thus considered to be outside the scope of ServiceCom.

Thirdly, reuse of the generated public interface of the composition is possible. The WSDL definition may be included in the web service provider library, making it available for use in other compositions. The interface file can also be published in for example an UDDI registry to make the composite service's functionality known to the outside world. For the latter purpose ServiceCom offers a deploy mechanism. This mechanism enables the publication of the new, composite web service in standard web service containers, such as Tomcat and J2EE.

2.3. Invocation phase

The *Invocation* phase deals with the execution of a web service composition and its corresponding result handling. In most cases users will likely want to incorporate the composition's functionality in custom made programs. In these situations the development of nice, user-friendly interfaces are left to the designers of these programs. However, some users may simply want to invoke the service once and are not interested at all in software program development. These users may employ the standard interfaces for the web service composition, which can be generated prior to invocation. Although these interfaces are basic, they provide a means to the user to enter the required inputs and display the received outputs. These interfaces can be reused in a similar fashion as the Java source and class files generated in the *Building* phase.

3. Demo

We will use a travel-planning scenario to illustrate the features of ServiceCom in the demo. The demo starts with a request from the user about the services he/she wants and the conditions, such as the ticket reservation, hotel booking, the dates, and price ranges, the preferences of the services, etc. The user will be presented with a set of dialog menus to specify, to select, and to modify, and to invoke. We will also demonstrate that the resulting composite web service can be published as a new service and its configuration is stored and can be re-used and extended. In summary we will demonstrate the following features of the tool: supporting an integrated environment for service composition, which includes composition specification, planning, implementation, and execution. Furthermore, we will demonstrate that ServiceCom supports composition publishing, re-use, and extension in a light-weighted manner, which enables it to be executed in different enterprise settings without too much implementation overhead.

4. Contributions

In this paper we gave a short description of ServiceCom, a light-weighted tool for service composition. This tool covers the complete life cycle of service composition, including the description, planning, building and executing of compositions. To the best of our knowledge there currently do not exist any other tools that provide such an integrated approach to service composition. Most importantly, because ServiceCom is based upon the service component mechanism, it promotes reusability and specialization in the specification of compositions. This sets it apart from other service composition tools based upon more traditional languages, such as BPEL4WS. Furthermore, in the building of service compositions reuse of class files is supported, thus shortening the building process increasing its efficiency.

References

[1] F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, S. Weerawarana; "Business Process Execution Language for Web Services", July 31, 2002, http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

[2] Business Process Modelling Initiative; "Business Process Modeling Language", http://www.bpmi.org

[3] Workflow Management Coalition; “XML Process Definition Language”, May 22, 2001, http://www.wfmc.org/standards/docs/ xpdl 010522..pdf

[4] F. Leymann; "Web Service Flow Language", 2001, http://www.ibm.com/software/solutions/webservices/pdf/WSFL.pdf

[5] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana; “ Web Service Description Language”, 15 March 2001, http://www.w3.org/TR/wsdl

[6] J. Yang, M.P. Papazoglou, W.J. v/d Heuvel; "Tackling the Challenges of Service Composition", *ICDE-RIDE workshop on Engineering E-Commerce/E-Business*, San Jose, USA, 2002

[7] J. Yang, M.P. Papazoglou; "Service Components for Managing the Life-Cycle of Service Compositions", *Information System*, June 2003 (forthcoming)

[8] C. Bussler; "Work Class Inheritance and Dynamic Work Class Binding", *Proceedings of the Workshop Software Architectures for Business Process Management at the 11th Conference on Advanced Information Systems Engineering (CAiSE*99)*, Heidelberg, Germany, 1999

[9] G. Kappel, P. Lang, S. Rausch-Schott, W. Retschitzegger; "Work Management Based on Objects, Rules, and Roles", *Bulletin of the Technical Committee on Data Engineering*, Vol. 18, No. 1, March 1995

[10] M.P. Papazoglou, A. Delis, A. Bouguettaya, M. Haghjoo; "Class Library Support for Work Environments and Applications", *IEEE Transactions on Computers*, Vol. 46, No.6, June 1997

Appendix A - ServiceCom screen shots

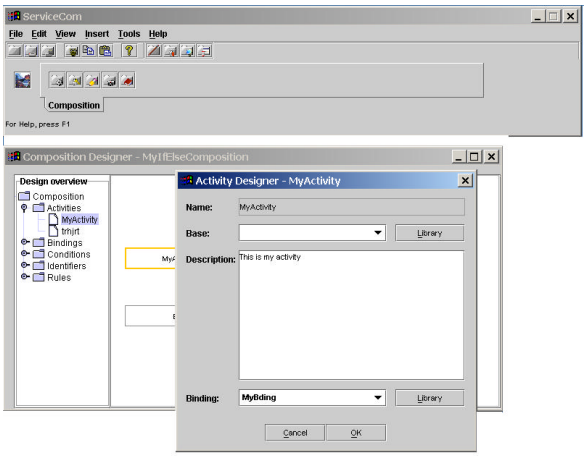

Figure 2. Main window, composition designer and activity designer

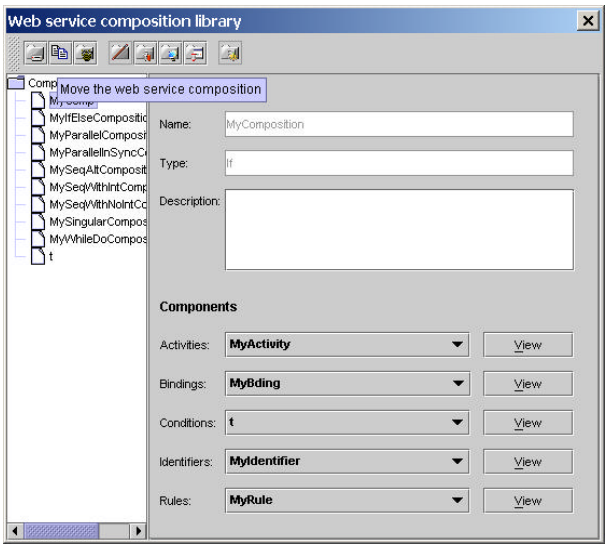

Figure 3. Web service provider library

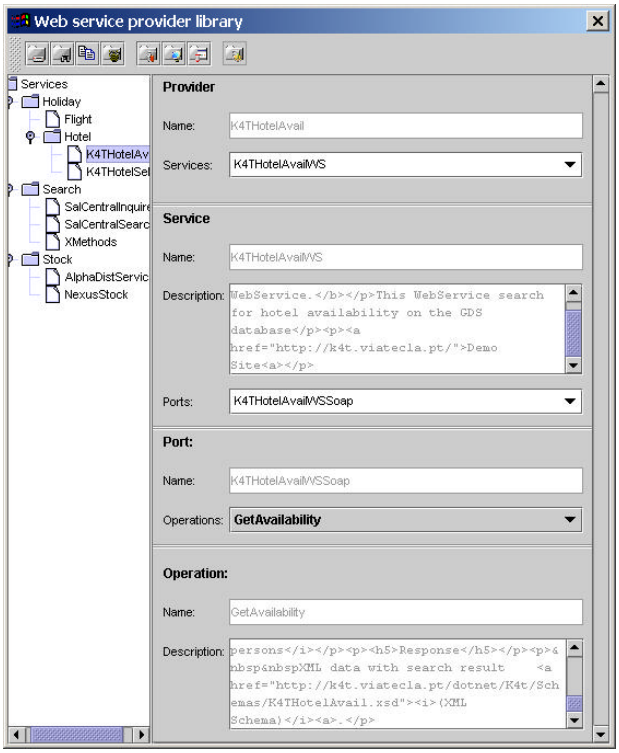

Figure 4. Web service composition library