

Dependency-aware Service Oriented Architecture and Service Composition

Jiehan Zhou^{1,2}, Daniel Pakkala¹, Juho Perälä¹, Eila Niemelä¹
Jukka Riekkö², Mika Ylianttila²

1. VTT Technical Research Centre of Finland
{firstname.surname@vtt.fi}

2. Department of Electrical and Information Engineering, University of Oulu
{jpr@ee.oulu.fi; mika.ylianttila@oulu.fi}

Abstract

Current service-oriented architecture (SOA) focuses on service composition for application development, i.e. application composition, which is perceived as volatile (i.e. “compose one time and use one time.”). This paper focuses on service composition information (i.e. service dependency) management and dependency-aware service composition for service development (i.e. service composition). This paper explores service composition and service dependency by proposing an extended SOA model, including: 1) establishing a dependency-aware service-oriented architecture (DSOA) that specifies dependency-aware service interactions, i.e. service publication, discovery, composition and binding; 2) developing an upper service dependency ontology for non-volatile DSOA service composition; and 3) demonstrating the validation of DSOA service composition by implementing a DSOA service manager.

1. Introduction

Service-Oriented Computing (SOC) is a new computing paradigm for service users to compose services for application and service development to experience the service age via Internet. Conventional SOC (e.g. service-oriented architecture [1], Web service description [2, 3] and service management platforms [4, 5]) focuses on service composition for application development (i.e. application composition), which is based on the following service interactions: service publication by service providers, service registry, and service discovery and binding by service requesters. As a result, the composition information between services will be lost after the services binding and execution, i.e. “compose one time and use one time.”

This paper focuses on dependency-aware service composition for service development (i.e. service composition) and service composition information (i.e. service dependency) management. The remainder of the paper is organized as follows. Section 2 defines concepts related to the DSOA model. Section 3 presents the motivation example specific to the home automation domain for addressing dependency-aware service composition. Section 4 presents DSOA architecture and an upper service dependency ontology is designed in Section 5. Section 6 preliminarily validates DSOA service composition paradigm. The conclusion and future work are presented in Section 7.

2. Definitions

There are various definitions about service with respect to different viewpoints[4, 6, 7]. We summarize characteristics of a service as follows: service is an action or operation directed by a service actor (e.g. a provider or requestor). Service is a marketing service in the sense of applying the marketing process: service advertisement, service discovery, and service engagement. Service is developed, deployed and invoked within a certain technical environment (e.g. operating systems and component standards). Service communicates with the environment through its interfaces, which encapsulate clear specifications of what the service requires and provides. Service resides in IP-capable devices; it can be remotely accessed and invoked via Web-accessible terminals. A service can be used for application composition and service composition.

Application composition is an abstract process of composing service descriptions. The composed application only becomes concrete at run time [8]. Composed applications do not provide interfaces for other services. In this sense, a composed application is volatile and cannot be deployed over Internet, which

5. Service dependency ontology

By analyzing the motivation example created in Section 3 and capturing main concepts with the bottom-up analysis method, we develop an upper service dependency ontology (Figure 2).

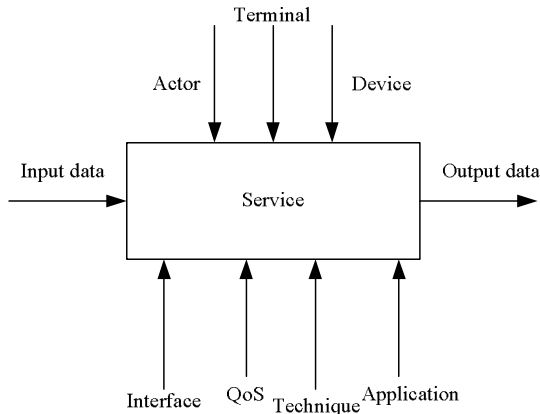


Figure 2. DSOA upper service dependency ontology

Actor-directed. Service must be directed by actors. An actor might be a person, organization, or a software agent that either seeks Web services or provides Web services.

Device-deployed. Service must be hosted and deployed by at least one device. For example, in the motivation example, the motion detecting service is hosted by a motion detection device.

Interface-provided. Normally, a service will have multiple interfaces corresponding to different access points.

Terminal-accessed. Service must be accessed by at least one terminal. Terminal examples are a standard laptop, Personal Digital Assistant, mobile phone, etc.

Data-processing service. Service must require and produce some types of data as its inputs and outputs. The type of each data may be a WSDL message type or an XML Schema simple type.

QoS-constrained service. Service must meet specified quality requirements to a certain degree. These quality requirements are so-called QoSs.

Technique-enabled service. Service must apply techniques or standards in the request and provision of the service. For example, the end user remotely requests the security notification service by applying networks of xDSL or Ethernet.

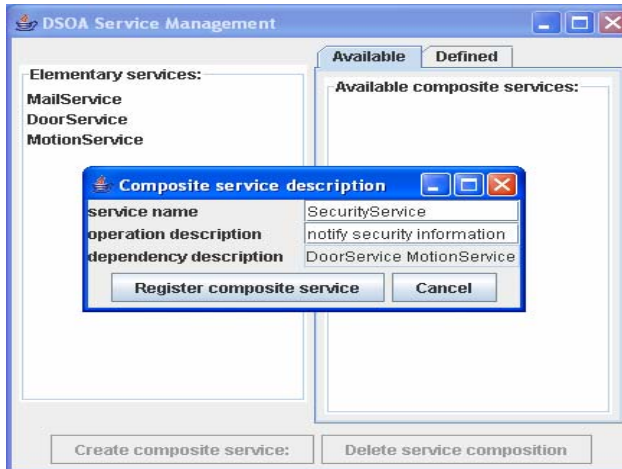
Application-oriented service. Application utilizes services and provides various user interfaces to enable the end user to achieve certain intentions, e.g. a home control application uses device services to control home devices, and home automation services to control

higher-level automation mechanisms such as complex alarms, dedicated robot presence, etc.

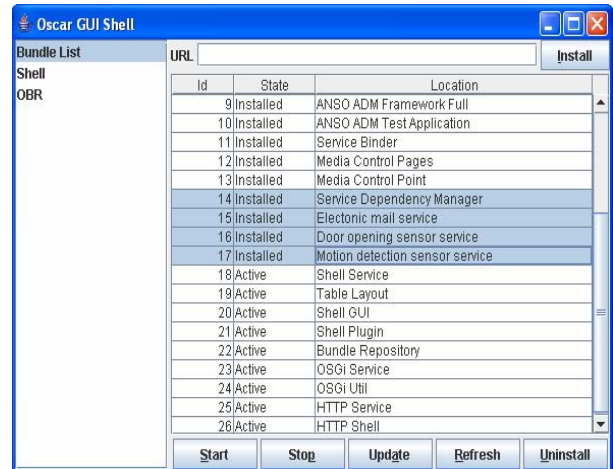
6. Implementation

The preliminary demo (Figure 3) implements a dependency-aware service manager, which aims to validate DSOA service composition, i.e. “compose once, use many times” and dependency-awareness. In a conventional way, the event of sending a security notification can be fired by opening door and detecting motion separately and consequently. In the demo, this event will be ignited by the composite service of security notification. The composition information about the service of security notification includes: the service of security notification uses door open service and motion detection service; the service of door open precedes the service of motion detection. In addition, the composite service of security notification will be defined and registered into the service manager for future use. To sum up, the implementation of DSOA service manager involves the following techniques:

- The preliminary DSOA service manager bundle provides a service composer with a platform for authoring composite services, describing composition dependency information, and registering composite services (Figure 3 (a)).
- Using the OSGi framework [4] for manipulating (e.g. registry, unregistry) elementary services (e.g. door open, motion detection) and the DSOA service manager bundle (Figure 3 (b)).
- Using the Oscar shell makes a service composer able to interactively and graphically access the OSGi services, e.g. activate and deactivate. Oscar is a compliant implementation of the OSGi framework specification [9] (Figure 3 (b)).
- The DSOA service manager allows a service composer to create composite services and specify composition dependency information, and to activate and deactivate the composite services based on the availability of elementary services.
- Complementary UPnP and OSGi Service framework. The UPnP device architecture specification and the OSGi Service Platform provide complementary functionality. The UPnP device architecture specification is a data communication protocol that does not specify where and how programs execute. That choice is made by the implementations. In contrast, the OSGi service platform specifies a (managed) execution point and does not define what protocols or media are supported. The UPnP specification and the OSGi specifications are fully complementary and do not overlap [4].



(a) Dependency-aware DSOA service manager and composite service description interface



(b) Access OSGi services with Oscar shell

Figure 3. DSOA service manager

7. Conclusions and future work

This paper describes a dependency-aware service-oriented architecture (DSOA) solution to the issue of service composition that allows the service composer to compose and register a service by analyzing and designing the dependency between existing Web services. DSOA supports service composition by: 1) establishing an extended SOA model, in which a role of a service composer, responsible for dependency-aware service composition, is introduced; 2) analyzing and designing service dependency ontology with a scenarios-based approach; 3) validating the concept of DSOA service composition by implementing a demo. This paper is the initial step for tackling the issue of service composition and service dependency. In future work we first plan to specify the DSOA model for service composition with more details (e.g., service interactions, service composition patterns, etc.). Secondly, we will extend the upper service dependency ontology for supporting complex DSOA service dependency description. Another key point is to further implement DSOA service management for deploying application composition and service composition in the field of home automation.

Acknowledgement

This work was funded by the Innovation in the ITEA-ANSO project, Tekes, the Finnish Funding Agency for Technology and Innovation, and VTT Technical Research Centre of Finland.

Special thanks to Timo Koskela and Juuso Ohtonen for their comments on the paper.

References

- [1] W3C-WSA, "Web Services Architecture," <http://www.w3.org/TR/ws-arch/#whatis>, 4 May, 2007.
- [2] W3C-WSDL, "WSDL: Web Services Description Language (WSDL) 1.1," <http://www.w3.org/TR/wsdl>, 4 May, 2007.
- [3] T. Andrews, et al., "Business Process Execution Language for Web Services (BPEL4WS), Version 1.1.," <http://ifr.sap.com/bpel4ws/index.html>, 4 May, 2007.
- [4] "OSGi Service Platform Specifications Release 4," http://osgi.org/osgi_technology/download_specs.asp?section=2#Release4, 4 May, 2007.
- [5] K. Arnold, "The Jini(TM) Specifications (2nd Edition)," Addison-Wesley, 2000.
- [6] J. Zhou and E. Niemela, "Beyond development-oriented software engineering: Service-Oriented Software Engineering (SOSE)," in *Service-Oriented Software System Engineering: Challenges and Practices*, Z. Stojanovic and A. Dahanayake, Eds. Hershey, USA: IDEA Group Publishing, 2005, pp. 27-47.
- [7] M. Fowler, "Inversion of Control Containers and the Dependency Injection pattern," <http://martinfowler.com/articles/injection.html#ConstructorVersusSetterInjection>, 4 May, 2007.
- [8] H. Cervantes and R. S. Hall, "Automating Service Dependency Management in a Service-Oriented Component Model," Proceeding of the 6th International Workshop on Component-Based Software Engineering - (CBSE), Portland, USA, 2003.
- [9] "Oscar: An OSGi framework implementation," <http://oscar.objectweb.org/>, 4 May, 2007.