Information and Software Technology 52 (2010) 304–323

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

A pattern-based approach to protocol mediation for web services composition

Xitong Li [a,*], Yushun Fan [a], Stuart Madnick [b], Quan Z. Sheng [c]

[a] Department of Automation, Tsinghua University, Beijing 100084, PR China
[b] MIT Sloan School of Management, 50 Memorial Drive, Cambridge, MA 02142, USA
[c] School of Computer Science, University of Adelaide, Adelaide, SA 5005, Australia

ARTICLE INFO

Article history:
Received 6 September 2008
Received in revised form 2 November 2009
Accepted 4 November 2009
Available online 11 November 2009

Keywords:
Service oriented architecture
Web service
Service composition
Protocol mediation
BPEL

ABSTRACT

Context: With the increasing popularity of Service Oriented Architecture (SOA), service composition is gaining momentum as the potential silver bullet for application integration. However, services are not always perfectly compatible and therefore cannot be directly composed. Service mediation, roughly classified into signature and protocol ones, thus becomes one key working area in SOA.
Objective: As a challenging problem, protocol mediation is still open and existing approaches only provide partial solutions. Further investigation on a systematic approach is needed.
Methods: In this paper, an approach based on mediator patterns is proposed to generate executable mediators and glue partially compatible services together. The mediation process and its main steps are introduced. By utilizing message mapping, a heuristic technique for identifying protocol mismatches and selecting appropriate mediator patterns is presented. The corresponding BPEL templates of these patterns are also developed.
Results: A prototype system, namely Service Mediation Toolkit (SMT), has been implemented to validate the feasibility and effectiveness of the proposed approach.
Conclusion: The approach along with the prototype system facilitate the existing practice of protocol mediation for Web services composition.

1. Introduction

Service Oriented Architecture (SOA) is a newly-emerging software architecture consisting of loosely-coupled services that communicate with each other through open-standard interfaces [1,2]. With the increasing popularity of SOA, service composition is gaining momentum as the potential silver bullet for the seamless integration of heterogeneous computing resources, rapid deployment of new business applications, and increasing reuse possibilities to a variety of legacy systems [3–5]. In real-world situations, however, independently-developed services are not always exactly compatible and cannot be straightly composed together.

An effective solution to these challenges is *service mediation*, which enables a service requester to connect to a relevant service provider regardless of the heterogeneities between them and works in a transparent way – neither of them needs to be aware of its existence [6,7]. First proposed in the Enterprise Service Bus (ESB) industry community [8], service mediation is referred to as the act of retrofitting existing services by intercepting, storing, transforming, and (re-)routing messages going into and out of these services [9]. Nowadays, service mediation has become a key working area in the field of SOA and Component-Based Software Engineering (CBSE) [10–12].

Service mediation can be roughly classified into *signature* and *protocol*. Signature mediation, which focuses on message types, has received considerable attention [13,14] and many commercial tools have been developed, such as Microsoft BizTalk Mapper,[1] Stylus Studio XML Mapping Tools[2] and SAP XI Mapping Editor.[3] In comparison, the problem of protocol mediation (also known as process mediation), which aims at reconciling mismatches of message exchanging sequences, is still open. A frequently-used approach to this problem is to develop a mediator/adapter which is a piece of code that sits between the interacting services and reconciles the mismatches [15,16]. However, the mediators developed by existing approaches have no control logics and cannot resolve complex mismatches. Few of these approaches can generate executable code of the mediators. Additionally, no existing approach provides a comprehensive solution to protocol mediation for Web services composition. Last but not least, there is a lack of user-friendly GUI tools that assist developers to alleviate their efforts on mediation

* Corresponding author.
E-mail address: lxt04@mails.tsinghua.edu.cn (X. Li).

[1] http://msdn.microsoft.com/en-us/library/ms943073.aspx.
[2] http://www.stylusstudio.com/.
[3] http://www.wsw-software.de/en-sap_services-mapping_sap_xi.mapping-sap-xi.html.

0950-5849/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2009.11.002

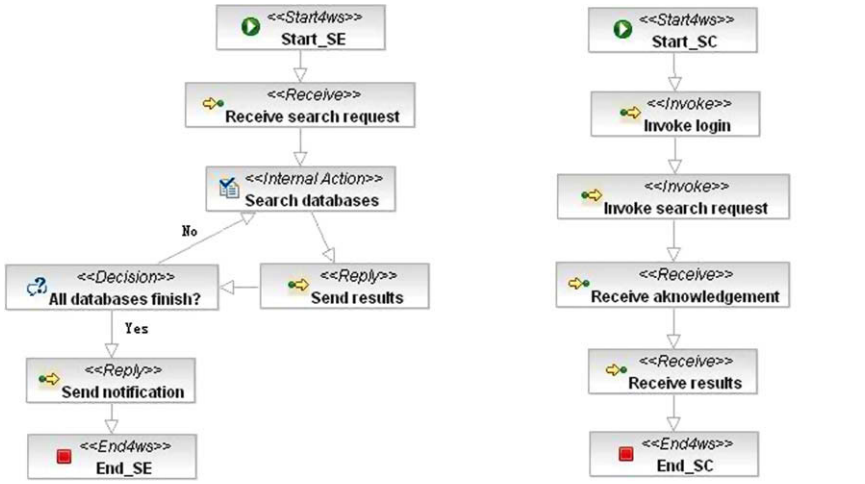

Fig. 1. A motivating example of service composition with protocol mismatches.

tasks, such as identifying protocol mismatches or generating mediation code. The work of this paper aims at addressing these issues.

1.1. Motivating example

We present a motivating example that will be used to demonstrate our research idea and approach throughout the paper, as shown in Fig. 1.

The example consists of a search client (S_C) and a search engine (S_E). S_C invokes S_E by sending its login information and the search request, respectively. Then, S_C waits for the acknowledgement and the results from S_E. On the other hand, after receiving search request, S_E starts to search several distributed databases one by one (by performing its internal searching action). Once S_E finishes a database and obtains some items, it sends these items to S_C immediately. In case all databases have been searched, S_E sends a completing notification to S_C and the search work is finished.

Among various specification languages of service composition (e.g., BPEL, WS-CDL, WSCI), BPEL obtains the dominance and has been proposed by OASIS as an industry standard. In this paper, we take BPEL as the specification language for describing service protocols. Fig. 1 shows the BPEL protocols of S_E and S_C. It is easy to see that S_E and S_C are partially compatible. They provide complementary functionalities but do not fit each other exactly. Apparently, without reconciling the protocol mismatches, S_E and S_C cannot interact with each other successfully.

1.2. Contributions

The rationale of our work has been presented in several conference papers [17–20]. As an extension of our previous work, we aim at developing a systematic approach to semi-automatically generating mediators for reconciling protocol mismatches. The main contributions are as follows:

(1) We present several basic mediator patterns which are derived from basic protocol mismatches. With the knowledge of protocol mismatches, the basic mediator patterns can be configured and composed by developers to construct advanced mediators and reconcile all possible protocol mismatches. Thus, the basic mediator patterns are referred to as a sufficient set of building blocks.
(2) We propose a technique to semi-automatically identify protocol mismatches when two partially compatible services need to be composed together. The technique is based on message mappings which need to be specified by developers. By using the technique, basic mediator patterns are selected according to the identified protocol mismatches.
(3) We develop BPEL templates for the mediator patterns which can be used to generate executable mediation code. Each mediator pattern has a corresponding BPEL template and a composite mediator corresponds to a combined BPEL-based mediation code.
(4) We propose a systematic engineering approach for service developers to reconcile protocol mismatches. The approach combines our work on identification of protocol mismatches, selection of mediator patterns and code generation of BPEL-based mediation.
(5) We develop a prototype system, namely *Service Mediation Toolkit* (*SMT*), which provides a user-friendly workbench and assist service developers to alleviate their mediation tasks. We use SMT to deal with several case studies in order to validate the feasibility and effectiveness of our approach.

The rest of the paper is structured as follows. In Section 2, several categories of basic protocol mismatches, referred to as basic mismatch patterns, and their composability are presented. In Section 3, basic mediator patterns are proposed to resolve the basic protocol mismatches. The configurability and composability of the mediator patterns are described in this section as well. The proposed solution to protocol mediation is presented in Section 4. The technique for selecting mediator patterns based on message mapping is also introduced and BPEL templates of the mediator patterns are developed for code generation of executable mediators. In Section 5, the prototype system SMT and the validation with a real-world case study are described. Related work is discussed in Section 6. Finally, Section 7 presents the conclusion and the future work.

2. Protocol mismatch patterns

2.1. Basic mismatch patterns

Protocol mismatches refer to such mismatches that occur between the message exchanging sequences of the interacting services. The existing work has identified several categories of protocol mismatches [21]. However, few paper claims its identification is complete in any sense.