Future Generation Computer Systems 29 (2013) 287–299

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

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

Assessing the replaceability of service protocols in mediated service interactions[☆]

Zhangbing Zhou[a,b], Walid Gaaloul[b], Lei Shu[c,*], Samir Tata[b], Sami Bhiri[d]

[a] *School of Information Engineering, China University of Geosciences, Beijing, China*
[b] *Computer Science Department, TELECOM SudParis, France*
[c] *Nishio Lab, Department of Multimedia Engineering, Osaka University, Japan*
[d] *Digital Enterprise Research Institute, National University of Ireland, Galway, Ireland*

ARTICLE INFO

Article history:
Received 28 September 2010
Received in revised form
28 March 2011
Accepted 5 August 2011
Available online 1 September 2011

Keywords:
Replaceability assessment
Mediated service interaction
Service protocol

ABSTRACT

Given the inherent autonomy, heterogeneity, and continuous evolution of Web services, mismatches usually exist between service protocols and mediated interactions are a common style of service interactions. Given a requestor service and an interaction to be conducted, if the provider service is found unavailable, we need to identify the most suitable provider service from a set of functionally equivalent candidates to replace the original one. Current techniques analyzing protocol replaceability can compute a replacement degree that specifies how replaceable two protocols are, but they cannot determine whether or not, and under which conditions, the effects prescribed by the requestor can be achieved. To address this challenge we propose a technique called *replaceability assessment* in this paper where, according to the adaptation mechanisms of a certain adapter, this technique (i) provides a set of condition pairs that determine when one protocol can be replaced by another, and (ii) computes a replacement degree. The set of condition pairs and the replacement degree are two complementary criteria to be used by the requestor for identifying the most suitable provider service.

1. Introduction

In service-oriented computing [1], independently developed software components are encapsulated as Web services, which interact with each other for achieving certain goals prescribed by requestors. Given two Web services, interactions between their protocols can be conducted (i) in a direct manner, called *direct service interactions*, as shown in Fig. 1(a), or (ii) in a mediated manner, called *mediated service interactions*, as shown in Fig. 1(b). *SP1* and *SP2* represent two service protocols. The symbols *A*, *B*, and *C* are the messages to be sent or received by *SP1* and *SP2*. Direct service interactions depend on the following assumptions: (i) service protocols consent to their message format and exchange sequences, and (ii) the composed process of service protocols can execute in a logically correct way (e.g., can terminate) [2,3]. Given the inherent autonomy, heterogeneity, and continuous evolution, Web services, which are complementary from a functional perspective, may be different in their protocols [2]. Hence, the assumptions of direct service interactions may not hold and service interactions are often conducted by means of adapters [2]. As shown in Fig. 1(b), an adapter acts as a centric *arbiter* [4] that reconciles mismatches and makes service protocols *compatible*. By *adaptation* we mean the act of identifying, classifying, and reconciling mismatches between service protocols [5].

Before or during conducting an interaction, the provider service may be found unavailable. Then, the requestor needs to find another provider service to replace the original one. Usually there are multiple provider services that can interact with a certain requestor service for achieving a certain goal. The requestor should select a provider service from a set of functionally equivalent candidates, which (i) can achieve the interactions that fulfill the requestor's requirements, and (ii) is the most suitable among the set of candidates. To achieve these, we propose a technique called *replaceability assessment*, which (i) provides a *set of condition pairs* that determine when two service protocols can be replaced, and (ii) computes a *replacement degree* that specifies to what extent one protocol is replaceable by another. The set of condition pairs and the replacement degree are two complementary criteria to be applied by the requestor for identifying the most suitable provider service. Based on the set of condition pairs, the requestor first filters candidate provider services by examining whether interactions fulfilling her requirements can be conducted. She then selects the most suitable provider service among the functionally equivalent candidates according to their replacement degrees. The selected provider service can interact with the requestor

☆ This paper was funded by French ANR funded research project (PAIRSE).
* Corresponding author.
E-mail addresses: zhangbing.zhou@gmail.com, zbzhou@cugb.edu.cn (Z. Zhou), walid.gaaloul@it-sudparis.eu (W. Gaaloul), lei.shu@ieee.org (L. Shu), sami.bhiri@deri.org (S. Bhiri).

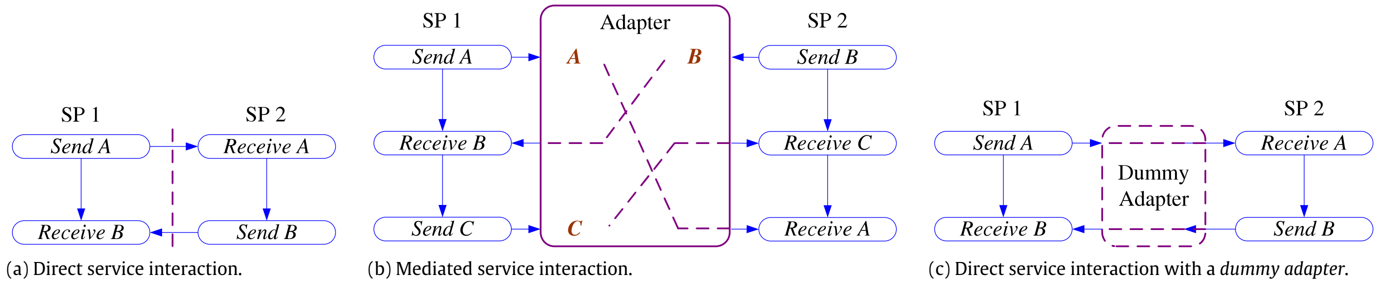

Fig. 1. Direct and mediated service interactions.

service for achieving the requestor's goal. In this paper we focus on assessing the replaceability in the context of mediated service interactions. Our technique can be applied to analyze direct service interactions as well. Indeed, direct service interactions can be regarded as special cases of mediated service interactions where, as illustrated by Fig. 1(c), a *dummy* adapter exists whose functionality is to forward messages between service protocols.

There exist several notions, like *similarity*, *substitutability*, *bi-simulation*, and *equivalence*, which specify the similar meanings as replaceability. Multiple techniques have been proposed for analyzing the replaceability of service protocols [6–12]. However, the context of these techniques is direct service interaction. Some techniques can compute a degree specifying how replaceable two service protocols are, but they cannot determine when one protocol can be replaced by another. Consequently, the selected provider service may not be suitable since interactions fulfilling the requestor's requirements may not be achieved using this provider service.

In this paper we assess the replaceability of service protocols as follows. We first abstract a service protocol by applying a set of rules. Control structures in a service protocol prescribe sequencing constraints between activities. These sequencing constraints specify prescribed message exchange orders. In mediated service interactions, message production and consumption are time-decoupled [2,13–15]. Indeed, *transformation rules* proposed by [16] suggest that some sequencing constraints may be relaxed for deciding if an *executable* and an *abstract* processes are compatible, although all sequencing constraints are respected at runtime. Depending on the adaptation mechanisms of a certain adapter, we propose a set of rules, which are used to reduce a service protocol into an *abstract service protocol*. Depending on the abstract service protocols, we construct a matrix called a *replacement matrix*, which specifies to what extent, as well as under which conditions, the *instances* of one abstract service protocol can be replaced by the *instances* of another abstract service protocol. Based on the replacement matrix, we assess the replaceability of service protocols by providing a set of condition pairs and computing a replacement degree.

The reminder of this paper is structured as follows. In Section 2, we introduce background concepts and present the mismatch patterns reconciled by the *general* adapter used in this paper. In Section 3, we present a motivating example which is used throughout this paper. In Section 4, we present a set of reduction rules which are applied for generating abstract service protocols. In Section 5, we construct the replacement matrix, and assess the replaceability, based on the abstract service protocols. In Section 6, we describe the prototype implementation. In Section 7, we review related techniques. We finally conclude this paper in Section 8.

2. Preliminaries

In this section we introduce the concept of service protocol, and present the mismatches resolvable by the *general* adapter *APT* used in this paper.

2.1. Service protocol

A service protocol specifies sequencing constraints between activities by means of control structures: *Sequence*, *Switch*, *Flow*, and *Loop*, in a recursive and nested manner. An activity sends or receives a message that contains business data. We assume that a service protocol does not contain duplicate activities. A service protocol may define conditions that guard transitions between activities. We denote *ACT* and *CD* as the set of activities, and the set of conditions, in a service protocol, respectively.

Definition 1 (*Service Protocol*)**.** A service protocol is a tuple $p = (MSG, ACT, CNT, DL, CD)$. $MSG = \{msg\}$ is a finite set of messages where a message contains a set of attributes. $ACT = \{act\}$ is a finite set of activities that send or receive these messages. $CNT \subseteq \{Sequence, Switch, Flow, Loop\}$ are control structures. $DL = \{dl\}$ is a finite set of directed links (i) connecting activities and control structures, and (ii) specifying sequencing constraints among activities [17]. $CD = \{cd\}$ is a finite set of conditions defined upon DL.

To facilitate the presentation of reduction rules in Section 4, we define a service protocol by adopting a block-oriented protocol model. Note that the approach presented in this paper is general and can be applied to a graph-oriented protocol model as well. Fig. 2 shows the grammar that defines service protocol models. p is the only non terminal in the block-oriented grammar. $act \in ACT$ represents an activity. $c_i \in CD$ represents a condition. Note that the control elements of defining a sequence: *StartSeq* and *EndSeq*, are optional.

2.2. Categorizing protocol mismatches

Mismatches between service protocols [18–20] can be classified into the following three categories:

1. *Attribute difference in messages*. This mismatch happens when the attribute sets of corresponding messages in different service protocols are not equivalent. Concretely, a message usually contains one or multiple attributes. We denote $ATTR(m)$ as the set of attributes in a message m. Given two messages m_1 and m_2 in two service protocols p_1 and p_2, this mismatch happens when $ATTR(m_1) \neq ATTR(m_2)$. Most adapters can reconcile this mismatch by splitting and merging messages.
2. *Sending and then receiving message mismatch*. This mismatch means that the messages to be received by a service protocol have been sent by partner service protocols previously, instead of at the current stage. Concretely, assuming two messages m_1 and m_2 are specified in both p_1 and p_2. This mismatch happens when p_1 sends m_1 and then receives m_2, but p_2 sends m_2 and then receives m_1. Most adapters can reconcile this mismatch by (i) temporarily saving m_1 for p_2 and m_2 for p_1 firstly, and (ii) forwarding m_2 to p_1 and m_1 to p_2 afterwards.
3. *Deadlock*. This mismatch means that all service protocols expect to receive messages, but these messages have not been sent