Analyzing Mediated Service Protocol Interactions Considering User's Expectation

Zhangbing Zhou[1], Han-Chieh Chao[2], Sami Bhiri[1], Lei Shu[3], Manfred Hauswirth[1]
[1]Digital Enterprise Research Institute, National University of Ireland, Ireland
[2]College of Electrical Engineering & Computer Science, National Ilan University, Taiwan
[3]Nishio Lab, Department of Multimedia Engineering, Graduate School of Information Science and Technology
Osaka University, Japan
zhangbing.zhou@gmail.com, hcc@mail.ndhu.edu.tw, sami.bhiri@deri.org
lei.shu@ist.osaka-u.ac.jp, manfred.hauswirth@deri.org

Abstract

A main promise of the service-oriented computing paradigm is to support a seamless interaction between service protocols. Given the inherent autonomy, heterogeneity, and continuous evolution of Web services, mismatches usually exist between service protocols. Hence, to efficiently identify and then select a suitable service protocol among functionally equivalent candidate service providers to achieve a certain business goal, an important criterion to the requestor is the knowledge about whether or not, and under which conditions, expected service interactions can be mediated.

In this paper we propose a formal technique to verify whether an expected service interaction is adaptable. We first present our observation that a mediated service interaction is synchronizable. Then, we formally model a protocol scenario (i.e., a part of a service protocol that can be enacted in an expected interaction) and an adapter, generate an adaptation logic, and formalize a mediated service interaction and its conversation. These enable one to perform a formal verification which examines whether or not, as well as under which condition, an expected interaction is achievable. The technique presented in this paper complements the efforts of adapter synthesization for ensuring the achievability of a certain expected interaction.

Keywords: Mediated service interaction, Service protocol, Formal verification, Expected behaviour.

1 Introduction

Efficiently discovering and then selecting a suitable provider service, which can interact properly with a requester service to achieve a certain business goal, is a fundamental promise of service-oriented computing paradigm. Given the inherent autonomy, heterogeneity, and continuous evolution of Web services, mediated service interactions, which are also called mediation-aided compositions [1], are a common style of Web service interactions [1]. Standards, such as BPEL [2] and WS-CDL [3], lay out the foundation that industry can build upon. Research efforts relating service interactions either (i) analyze service compositions [4-5] that target direct service interactions, or (ii) detect and correct deadlocks (or mismatches) in service choreographies [6], or (iii) synthesize adapters [1][7-9] that facilitate mediated service interactions through identifying, classifying, and reconciling mismatches. These techniques support service interactions from a global behaviour perspective without considering the requestor's expectations. Specifically, for two service protocols of a requestor and a provider, they usually can conduct many (direct or mediated) service interactions, but according to the requestor's requirement, a few interactions (in percentage) are interested whereas the others are not relevant. For instance in the motivating example (see Section 1.1), assuming that only the interaction leading *Toy Items* to be delivered is expected. Then, the criterion to select a suitable service provider is the support of these expected interactions, whereas the other interactions are complementary but not mandatory. To achieve this selection criterion, we propose a formal technique which verifies whether or not, and provides a condition that determines when, an expected interaction can be mediated [10].

Techniques that analyze direct service interactions can somehow be applied to study mediated service interactions. However, different from a service protocol which is autonomous, an adaptation logic is coupled with participating service protocols, and a mediated service interaction often adopt a *hub-and-spoke* topology [11] and hence is synchronizable [4] (see Section 3). These characteristics facilitate the verification of mediated service interactions.

As reviewed in [10], several efforts synthesize adapters. However, a technique that synthesizes an adapter does not provide the level of evaluation we target. The existence of an adapter indicates that there are some interactions possible between two service protocols, but an adapter itself does not inform the requestor about whether an expected interaction is supportable, although this knowledge may be derived by simulating the message exchange through the adapter. Moreover, an adapter does not prescribe the

*Corresponding author: Zhangbing Zhou; E-mail: zhangbing.zhou@gmail.com

condition that determines when an interaction is possible. Hence, even if an expected interaction is achievable from the message exchange perspective, it may fail due to the unsatisfiability of some conditions with respect to exchanged message instances.

1.1 Motivating Example

Figure 1 depicts three service protocols including two toy shop services (denoted TS_1 and TS_2) and one toy requestor service (denoted *REQ*). *Rec*., *Rep*., and *Inv*. means *receive*, *reply*, and *invoke* respectively. We use abstract BPEL to represent service protocols since BPEL is widely accepted by industry and academia as the de facto standard to model Web services based protocols. *Switch* and transition conditions (i.e., Cd_i ($i \in [1,9]$)) are associated with links. *Switch* conditions refer to the conditions specified on conditional branches in *Switch* blocks.

The difference between TS_1 and TS_2 is that, TS_1 may apply a discount on the price depending on *Cust*. *Info*. Hence, a *Cust*. *Info*. is expected before the price is decided. A *Normal Price* is always applicable by default. On the contrary, TS_2 can provide a price if the requestor is confirmed to be an adult. In addition, some conditions specified on the links of TS_1 and TS_2, such as Cd_1 in TS_1 and Cd_6 in TS_2, are different.

Without loss of generality, we assume that a requestor, using *REQ*, chooses from TS_1 or TS_2 to interact with, to buy some toys online as a gift for her kid. The expected result is *Toy Items* being delivered. Hence, she is eager to know which (either TS_1 or TS_2) is a suitable candidate service provider to achieve this expected interaction.

We first examine the support of the expected interaction between TS_1 and *REQ*. Due to privacy concerns, *REQ* sends *Cust*. *Info*. if she is convinced by the price and is committed to buy. Consequently, a deadlock occurs when TS_1 requests *Cust*. *Info*. before sends the *Price*, while *REQ* expects the *Price* before decides upon continuation and sends *Cust*. *Info*. or not. Such (kind of) deadlock is reconcilable according to the adaptation mechanisms of [8] (through providing missing *Cust*. *Info*. using *evidences*) and [9] (through generating *mock-up Cust*. *Info*. message), but is beyond the capacity of other adapters [1][7][12] since they consider any deadlock as an unreconcilable mismatch.

We then examine the support of the expected interaction between TS_2 and *REQ*. As discussed in the above paragraph, [8] can support it through providing a missing *Cust*. *Info*. using *evidences*, although this *Cust*. *Info*. may not be consistent with the interaction context since it may be different from what is provided by *REQ* afterwards (through *Inv*. *Cust*. *Info*. activity). On the other hand, the deadlock is unresolvable according to [9] since Cd_6 cannot be enabled using a mock-up *Cust*. *Info*. message.

From the reasoning above, the requestor got to know that TS_1 supports the expected interaction using an adapter of [8-9], but TS_2 supports it using an adapter of [8] only.

Naturally, the requestor is also interested in the fact that whether she can cancel the interaction if the price is unacceptable. According to the reasoning above, through an adapter of [8-9], both TS_1 and TS_2 can support this expected interaction that leads both TS_1 (or TS_2) and *REQ* to their cancellation.

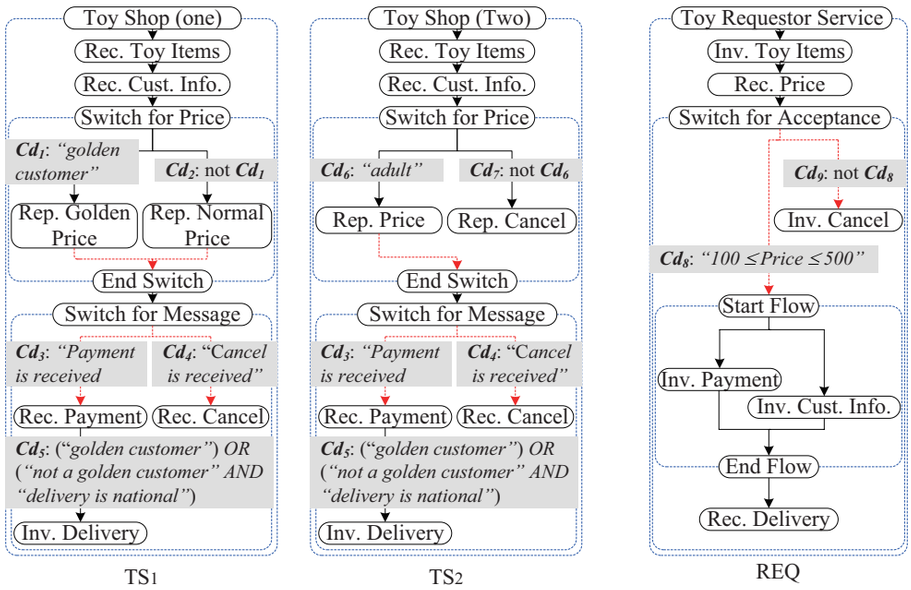

Figure 1 Service Protocols for Two Toy Shop Services (i.e., TS_1 and TS_2) and One Toy Requestor Service (i.e., *REQ*). A Requestor, Using *REQ*, Can Choose from Toy Providers TS_1 or TS_2 to Interact with, for Purchasing Some Toys Online

Consequently, the requestor is informed that TS_1 and TS_2 are both suitable if an adapter [8] is used, while TS_1 is more suitable if an adapter [9] is used. An adapter such as [8-9] can specify whether an interaction is possible, but it cannot indicate whether a specific (maybe also an expected) interaction is supportable.

The discussion above explores possible message exchanges between service protocols. Indeed, the success of an adaptation in particular, and an interaction in general, depends on *conditions* that decide which branches to follow in *Switch* blocks and guard transitions between activities. For instance, TS_1 and *REQ* are adaptable such that *Toy Items* can be delivered. A prerequisite is that the condition $Cd_2 \wedge Cd_3 \wedge Cd_5 \wedge Cd_8$ is satisfiable, such that, Cd_2, Cd_3 and Cd_8 lead TS_1 and *REQ* to choose their desired branches, while Cd_5 ensures satisfying specific business requirements. Identifying these conditions is beyond the capability of adapter synthesization techniques.

Other factors, such as functional properties (normally IOPE [13-15]) and non-functional properties (QoS [16-17], security [18-19], reputation [20], and others), are also important to be considered in the service discovery and selection procedures. These factors are complementary to our work and are outside the scope of this paper. We focus on service protocol adaptability analysis in this paper.

The verification of whether or not, and the identification of the condition that determines when, an expected interaction is supportable, is beyond the capability of existing techniques that analyze service protocol interactions. In the following, we show how they are explored in this particular example and in general.

1.2 Approach

In this section, we first clarify the research context, and then introduce the approach.

1.2.1 Context

There are two types of method that study service interactions: (i) proposing user-defined techniques like [12], or (ii) using techniques like model checking to verify certain properties of a service interaction. We call the latter strategy a formal approach where, service interactions are formally modeled, and properties to be verified are usually specified in terms of linear temporal logic (LTL) formulae. Then, the verification can be achieved automatically. Since model checking methods trace through all relevant states with respect to a given property which refer to all possible execution paths, the verification is more thorough and efficient. Considering these, we choose a formal approach in this paper.

Verifying service interactions formally is an active research area and several methods, such as [4-5][21-22], have been proposed. [4] proposes an inspiring method to study interacting BPEL processes, which identifies sufficient conditions that determine when the conversation set for bottom-up specified service compositions remains the same for synchronous and asynchronous communication semantics. We adopt the technique proposed in [4] to perform our verification, thanks to our observation that a mediated service interaction is synchronizable. Another major concern is that we can reuse the tool developed in [4].

We conduct the verification in accordance to our Space-based Process Mediator (SPM) [9]. Note that the technique proposed in this paper is general and can be applied to other adapters as well. In a nutshell, our SPM can reconcile the deadlock in our motivating example in the *Rec*. *Cust*. *Info*. activity by providing a *mock-up Cust*. *Info*. message, and this *mock-up Cust*. *Info*. message is replaced when the *Cust*. *Info*. message is provided by *REQ* to enable the *Inv*. *Delivery*. Generally, the SPM can circumvent this kind of deadlock, if the dependency that exists between one of the receiving activities and one of its direct-succeeding activities is neither control nor mandatorily data dependent. Then, the SPM can produce a *mock-up* message that acts as the data expected by this receiving activity. This *mock-up* message is replaced by a concrete message when it is provided by a peer protocol.

1.2.2 Approach

After introducing the preliminaries, we present in Section 3 an observation that a mediated service interaction is synchronizable. This is a prerequisite to our formal verification since nonsynchronizable service compositions may cause the undecidability of LTL verification [4].

Then, in Section 4, we formalize a protocol scenario and an adapter in terms of Guard Finite State Automata (GFSA) [4]. BPEL is not suitable as a service protocol modeling method in formal approaches, although it is widely used in both academia and industry. GFSA is a finite state automata with guards specified on transitions [4][23], where guards correspond to conditions in BPEL specification. As such, we use the notions of guard and condition interchangeably. A protocol scenario is a part of a service protocol, which can be enacted in a particular (maybe expected) interaction depending on the evaluation of conditions in *Switch* branches. We study mediated service interactions at a protocol scenario level since a protocol scenario can represent the part of a service protocol involved in an expected interaction. The conjunction of transition guards in these interacting protocol scenarios constructs a condition, which determines when these service protocols can perform the expected interaction.

Afterwards, an adaptation logic with respect to interacting protocol scenarios, which is also formalized using GFSA, is generated. The adaptation logic is coupled with the specific protocol scenarios, although the adaptation mechanism of an adapter (like the SPM) is general.