Detecting feature interactions in Web services with model checking techniques

Abstract As a platform-independent software system, a Web service is designed to offer interoperability among diverse and heterogeneous applications. With the introduction of service composition in the Web service creation, various message interactions among the atomic services result in a problem resembling the feature interaction problem in the telecommunication area. This article defines the problem as feature interaction in Web services and proposes a model checking-based detection method. In the method, the Web service description is translated to the Promela language—the input language of the model checker simple promela interpreter (SPIN), and the specific properties, expressed as linear temporal logic (LTL) formulas, are formulated according to our classification of feature interaction. Then, SPIN is used to check these specific properties to detect the feature interaction in Web services.

Keywords feature interactions, web services, model checking, detection

1 Introduction

It is well known that the feature interaction problem is one of the important bottlenecks for the development of new telecommunication services. In the telecommunication area, feature is defined as a “unit of one or more telecommunications or telecommunications management based capabilities that a network provides to a user” by Bellcore. The feature interaction problem, first coined in the early 80’s by Bellcore, mainly includes interactions that occur because the requirements of multiple features are not compatible, and the interactions that occur when a feature behaves differently in the presence of other features [1]. A typical example is the interactions between call waiting and call forwarding when busy, both of which are subscribed by the same user. The related research study can be found in Refs. [1−5].

As a platform-independent software system, a Web service is designed to offer interoperability among diverse and heterogeneous applications. However, sometimes an individual Web service cannot fulfill the complex and increasing user demands. Thus, service composition, a technique to incorporate and orchestrate different individual services, is introduced to adapt to various user preferences and contexts. However, service composition results in a great deal of frequent and complex message interactions, which lead to the potentiality of similar problems in the Web services domain. Here, we define the problem as feature interaction in Web services, the problem resulting from the abnormal message interactions among the composed services in the service composition scenarios. In this article, we analyze the problem and propose a detection method with model checking techniques, one of the promising verification approaches.

The remaining sections of this article are organized as follows: Section 2 introduces the feature interaction in Web services. Section 3 provides a short description of the model checking technique. Section 4 describes our detection method in detail. Finally, Sect 5 concludes this article and mentions the future research.

2 Feature interactions in Web services

The feature interaction problem has become a major roadblock to implement new services in the telecommunication area. With the application of service composition techniques in the Web services domain, feature interaction in Web services also becomes one of the obstacles of new service creation. It may negatively impact the service quality and robustness, reduce the service scalability, and even lead to the failure of service composition in practice.

Since service creation in Web services is based on a more open, flexible, and diverse method, feature interactions in Web services are quite different from those in telecom in the following points:

1) Feature abstract is an important problem in the Web services domain. The clarity of telecom feature abstraction based on the functional nature greatly advances the related research. However, it is different in the Web services domain.
We cannot clearly pick out a feature as originating call screening or call forwarding in telecom, which increases the difficulty in the interaction scenario analysis.

2) The distributed nature of Web services leads to the privacy issue of atomic service, that is, the atomic service logic is hardly obtained except by the service provider. Thus, various static methods employed for the feature interactions in telecom cannot apply to the research on the feature interactions in Web services.

3) The feature interaction problem in the Web services domain is far more complex and serious because the numbers and types of web services are considerably greater than those of telecommunication services. In our point of view, the further research of the feature interaction problem in Web services must mainly include what its new characteristics are, especially its distinctions from feature interactions in other areas; how to classify this problem and illustrate the corresponding scenarios, and so on.

There has been little research on this problem, although it was raised in the sixth international workshop on feature interactions in telecommunication and software systems (FIW’00). Weiss proposed an approach, based on goal-oriented analysis and scenario modeling, for detecting feature interactions among Web services in Ref. [6]. He employed goal-oriented requirement language(GRL) to model the interactions between services, and analyzed and distinguished explicitly between functional and non-functional feature interactions. Besides, Wohlstadter E  et al. [7] presented a middleware GlueQoS to support the compatible quality of service(QoS) feature composition in Web services. On the basis of the analysis of the interactions among the QoS features, the GlueQoS policy mediator is used to handle them. The study is of help to detect and resolve the non-functional feature interactions in Web services.

3 Model checking technique

Model checking, a promising verification technique, was developed independently in the early 1980’s by Clark and Emerson and by Queille and Sifakis. It is defined as “an automated technique that given a finite-state model of a system and a logical property, systematically checks whether this property holds for (a given initial state in) that model” in Ref. [8]. In the general approach of model checking, there are three main elements: the modeling language to describe the system, the specification language to formulate properties to verify, and the verification algorithm.

The model checking technique has been applied to the Web services domain. Nakajima S [9–11] applied model checking to verify the properties of Web service flow aiming to raise the reliability of Web services. Kazhamiakin R [12] described a model checking-based approach for the formal specification and verification of distributed processes in a Web service framework, and Fu Xiang [13, 14] presented an analysis tool for the design composite Web services.

4 Method to detect the feature interactions in Web services

Since the feature interaction problem is very complex, it is essential to adopt a divide-and-rule approach to detect and resolve it, that is, a sound classification will be useful toward a general scheme for detecting or resolving feature interactions. We classify the feature interactions into five categories as deadlock, loop situation, invocation error, race condition, and resource contention.

This section describes our detection method in detail. Our method, based on the model checking technique and the model checker SPIN [15], aims to detect each type of feature interaction defined in our classification.

4.1 Model checker SPIN

SPIN is an automaton-based verification system for models of distributed software systems. It has been widely used to detect design errors in applications ranging from high-level descriptions of distributed algorithms to detailed code for controlling telephone exchanges. It adapts Promela as the input language and verifies the specific properties using exhaustive exploration of the system state space. It also employs the partial order reduction technique to reduce the state explosion problem.

Promela is used to construct models that the SPIN system can analyze. It is a specification language for distributed systems, which offers communication and concurrency primitives inspired by process algebras. It consists of sequential processes, variables and communication channels that are used for connecting sequential processes.

LTL is used to represent the proposed system properties in the SPIN system. It is an extension of propositional logic for expressing relationships between the orders of events occurring over time. LTL formulas are constructed from atomic propositions, logical operators $\land$, $\lor$, and $\neg$, and four temporal operators $X$ (next time operator), $G$ (globally operator), $U$ (until operator), and $F$ (finally operator).

4.2 Overview of our method

Our method (see Fig. 1) is based on the traditional model checking approach. We adopt SPIN as the verifier to check whether the feature interaction problem exists in our Web
services system. Firstly, we translate the Web services description into the input language of SPIN – Promela. Secondly, we provide a series of LTL formulas to express our different verification tasks according to our classification in Sect. 2. Thirdly, we obtain the verification results of SPIN with the Promela and the LTL formulas as the input. Fourthly and lastly, we conclude by mentioning the category of feature interaction that exists or that no feature interaction exists in the composite Web services after the analysis of the verification results.

Fig. 1 Model checking-based method

4.3 BPEL4WS to promela

We choose business process execution language for Web services (BPEL4WS) [16] to describe the orchestration processes in the service composition. BPEL4WS is a process modeling language developed by IBM, Microsoft, and BEA. It supersedes Microsoft’s XLANG and IBM’s Web services flow language (WSFL) [17]. In this language, activities are defined as the basic components of a process definition. There are two types of activities: basic activities and structured activities.

SPIN models specified in Promela consist of three types of objects: processes, message channels, and global variables. Processes specify behavior, whereas channels and global variables define the environment in which the processes run.

The translation templates of the basic activities in BPEL4WS to Promela are illustrated in Table 2.

<table>
<thead>
<tr>
<th>BPEL4WS</th>
<th>Promela</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic activity</td>
<td>Process</td>
</tr>
<tr>
<td>PartnerLink</td>
<td>Channel</td>
</tr>
<tr>
<td>Variable</td>
<td>Variable</td>
</tr>
</tbody>
</table>

Structured activities prescribe the order in which a collection of activities take place, mainly including sequence, switch, while, flow, and pick. We convert the BPEL4WS process with all structured activities to one Promela process. The structural constructs and their corresponding Promela program are illustrated in Table 2.

<table>
<thead>
<tr>
<th>BPEL4WS structural constructs</th>
<th>Promela program</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;sequence&gt;</td>
<td>proctype &lt;name&gt;( ) { ... run proc1( ); run proc2( ); ... }</td>
</tr>
<tr>
<td>&lt;case&gt;</td>
<td>if condition1=&quot;true&quot;&gt; ... run proc1( ) ... condition2=&quot;false&quot;&gt; run proc2( )</td>
</tr>
<tr>
<td>&lt;switch&gt;</td>
<td>{ ... case condition1=&quot;true&quot;&gt; :c == true&gt;&gt;&gt; run proc1( ) ... case condition2=&quot;false&quot;&gt; :c == false&gt;&gt;&gt; run proc2( ) }</td>
</tr>
<tr>
<td>&lt;while&gt;</td>
<td>{ ... do condition1=&quot;true&quot;&gt; :c == true&gt;&gt;&gt; run proc1( ) ... condition2=&quot;false&quot;&gt; :c == false&gt;&gt;&gt; break od; ... }</td>
</tr>
<tr>
<td>&lt;pick&gt;</td>
<td>proctype &lt;name&gt;( ) { ... do flow { ... run proc1( ); run proc2( ) }</td>
</tr>
<tr>
<td>&lt;onMessage&gt;</td>
<td>do onMessage { activity1 }</td>
</tr>
<tr>
<td>&lt;onAlarm&gt;</td>
<td>do onAlarm { activity2 }</td>
</tr>
<tr>
<td>&lt;flow&gt;</td>
<td>do flow { atomic{ run proc1( ); run proc2( ) } }</td>
</tr>
</tbody>
</table>

Here, we pay attention only to the part of the behavioral aspects in BPEL4WS. The other parts, for the sake of brevity, are not discussed.

4.4 LTL formulas for the feature interaction in Web services

According to our feature interaction classification, we express the properties to check as LTL formulas. Here, we illustrate the formulas with a concrete case.

In the example shown in Fig. 2, the Stock Trading Service implements a stock buying and selling mechanism. An upper and lower limit, which determines stock trading, is set in the service. Stock shares are sold when the upper limit is exceeded and shares are purchased when the quoted price is less than the lower limit. In the Stock Quote Service, the stock quote updates at intervals. Then, the new quote is sent to the Stock Trading Service. After the analysis of the stock quotes, the Stock Trading Service interacts with Bank Services to
To keep the stock trading system alive, we must not only fulfill the execution of the program code, but also ensure that the system operates properly according to our requirements. Then, we illustrate our formulas corresponding to the following types of feature interactions. Although some properties can be checked using the SPIN’s built-in ability, we still include these in our formulas. In the formulas, a (or similar alphabetic symbols) represents an activity in BPEL4WS.

**Property 1** Deadlock
That is: \( a \rightarrow \neg a \). This formula indicates that once an activity is initiated, the process of this activity will not fall into an endless state.

**Property 2** Loop situation
That is: \( \neg F(a \rightarrow Xb) \wedge (b \rightarrow Xa) \). This formula avoids the dead loop consisting of two (or more) activities.

**Property 3** Invocation error
That is: \( G(a \rightarrow F(b \vee c)) \). This formula ensures that the activation of an activity leads to desired following activities on the basis of the fulfillment of safety properties. The example of \( G(UpdateQuote \rightarrow F(BankTransaction \vee WaitQuoteUpdate)) \) indicates that once the stock quote is updated, the activity BankTransaction in the Bank Service or WaitQuoteUpdate in the Stock Trading Service must be activated later. There are some similar formulas such as \( G(\neg a \wedge b \rightarrow Fc) \) and \( G(\neg a \rightarrow b \rightarrow cXa) \).

**Property 4** Race condition
That is: \( G(a \rightarrow X \neg a \wedge \neg a \) ) (or \( a \vee (\neg a \) )). For instance, \( G(UpdateQuote \rightarrow X \neg UpdateQuote \wedge \neg UpdateQuote \cup (BankTransaction \vee WaitQuoteUpdate)) \) avoids the race condition between the activity UpdateQuote in the Stock Quote Service and the activity BankTransaction in the Bank Service. Once the Stock Trading Service determines to open a bank transaction, the activity BankTransaction must be fulfilled before the next UpdateQuote, otherwise, it may cause a mismatch between the trigger price and the transaction price. The similar formulas include \( Fa \rightarrow (((\neg a \cup b) \rightarrow ((a1 \wedge b1) \cup (a2 \wedge b2)))) \), etc.

**Property 5** Resource contention
That is: \( G(\neg (a \wedge b)) \). In this formula, it is assumed that there exists the possibility for the activities \( a \) and \( b \) to compete for the same resource simultaneously.

In the last step, we can judge whether feature interaction in Web services exists in the composite service according to the verification results of SPIN. We obtain two errors from the verification results of our example (see Fig. 2): one is a deadlock state, which proves to be a redundant state and can be deleted from the design of the composite service. The other is the race condition between the two activities: UpdateQuote and BankTransaction. As discussed above, there exists the possibility of mismatch between the trigger price and the transaction price because of the nondeterministic execution order of the two activities. We finally add the CancelTransaction activity in the Stocking Trading Service and the Refresh activity in the Bank Service to avoid the occurrence of this error.

### 4.5 Discussions

The study described above undoubtedly promotes the progress of research on the feature interaction in Web services. However, the proposed detection methods only focus on the non-functional feature interactions. Our method, based on the model checking techniques, can be used to effectively detect the functional feature interactions, the important part of feature interaction in Web services. Moreover, we believe that our detection method is a practical method in the specific application scenarios.

Service composition is the key cause of feature interaction in Web services. As already known, choreography and orchestration are the two key aspects of service composition. Choreography usually describes the service interactions from a global perspective, and is quite different from orchestration, which implies a centralized control mechanism. Moreover, the choreography language (such as WS-CDL) mainly uses state transition to express the internal logic and the control logic of atomic Web services. It cannot express the business process that a single party executes. Since our study focuses on the orchestrated Web services, it is difficult to apply our detection method to the composite Web services described in the choreography mode.

### 5 Conclusions and future works

This article mainly makes the following contributions to the study on the feature interaction in Web services. We present a classification of the problem and propose a model checking-based detection method. The powerful detection ability of model checking techniques is of great help to detect the feature interaction in Web services systems. Moreover, we make our analysis on the essence of the feature interaction in Web services to further the understanding of this problem. We assume that our work will help to further the research on the static detection of feature interaction in Web services,
especially the functional feature interactions.

Undoubtedly, there is still considerable research that remains to be done in this research area. For example, the investigation of the interaction scenarios in practice is required for an in-depth analysis of this feature interaction problem. Besides, the objective of feature interaction detection is to resolve the problem. Then, effective resolution methods remain to be explored.

Acknowledgements This work is supported by the National Natural Basic Research and Development Program (973 program) of China (2003CB314806); the Program for New Century Excellent Talents in University (NCET-05-0114); the Program for Changjiang Scholars and Innovative Research Team in University (PCSIRT).

References


Biographies: ZHANG Jian-yin, Ph. D. Candidate, State key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, interested in the next generation Internet service and telecommunication service.

YANG Fang-chun, Ph. D. professor and vice president, Beijing University of Posts and Telecommunications, interested in the telecommunication network technologies and communication software.

SU Sen, Ph. D., professor, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, interested in softswitch technology and next generation Internet service.