A Classification of Service Composition Mismatches to Support Service Mediation

Xitong Li, Yushun Fan, Feng Jiang
Department of Automation, Tsinghua University, Beijing 100084, China
lxt04@mails.tsinghua.edu.cn, fanyus@tsinghua.edu.cn, jiangf00@mails.tsinghua.edu.cn

Abstract

On building a service-oriented architecture for Grid systems, more and more researches have been focused on service composition. Services, however, are not always exactly compatible and much effort has to be addressed to mediate incompatible services, which is an unavoidable problem and recognized as service composition mismatches. To address it, a classification of composition mismatches is significant and helpful to divide the whole problem into different sub-problems which require corresponding mediators to deal with. However, there exist very few classification approaches specific to such problem and a comprehensive classification is still lacking. In this paper, a classification of most kinds of composition mismatches is introduced. Besides that, several basic protocol mismatch patterns are presented, which can be viewed as basic constructs of existing protocol mismatches. Finally, conclusions and the future work are drawn up.

1. Introduction

On building a service-oriented architecture for Grid systems, more and more researches have been focused on service composition. And the evolution of Grid systems requires a loosely-coupled environment that enables flexible composition of pre-produced services. Services, however, are not always exactly compatible and much effort has to be addressed to mediate these pre-produced but incompatible services. Recently, *service mediation* has become the status of a definite working area in the field of software engineering [1].

The incompatibility of services composition, namely *service composition mismatches*, has been an unavoidable and crucial problem. However, there exist very few classification approaches specific to such problem and a comprehensive classification is still lacking [2]. Moreover, the existing classifications are not sufficiently dedicated to the development of service mediators. In this paper, the authors expect to present a comprehensive and systematic classification of the problem space, which is helpful to thoroughly understand and conquer it by dividing it into several sub-problems. The benefit of dividing into sub-problems is based on the recognition that different kinds of mismatches require corresponding mediators to deal with. Another contribution of this paper lies in the introduction of six basic protocol mismatch patterns, some of which have not been identified before. And we believe all existing protocol mismatches can be composed from these six basic mismatch patterns. Note that the focus of this paper is the development of a classification of mismatch problems. A deeper research of the systematic engineering solution to the classified problems is beyond the scope of this paper and subject of future work.

The rest of this paper is structured as follows. In Section 2, a classification of service composition mismatches is presented and each category will be respectively introduced in detail, especially signature and protocol mismatches. Next, related work and comparisons with the work of this paper are presented in Section 3. Finally, conclusions and the future work will be drawn up in Section 4.

2. Classification of Service Composition Mismatches

There exist only a few approaches to classify composition mismatches. And most of the existing approaches just broadly distinguish between signature, protocol, semantic and non-functional mismatches from one-dimension classification. However, this simple one-dimension classification is not clarifying and some categories should be considered from different perspectives.

The Sixth International Conference on Grid and Cooperative Computing(GCC 2007)
0-7695-2871-6/07 $25.00

Table 1. A Classification of Service Composition Mismatches

	Functional		Non-Functional
	Signature	Protocol	
Syntactic (Structure)	Mismatches occurring at the structure of a method, such as parameters and data types, etc.	Mismatches occurring at the behavioral logic of interfaces	Mismatches occurring at the physical Quality of Service (QoS)
Semantic	Mismatches occurring at the semantics of the signature of a method	Mismatches occurring at the semantic definitions of protocols of interfaces	Mismatches occurring at the concepts of Quality of Service (QoS)

To get a deep insight of different kinds of mismatches, a comprehensive classification of service composition mismatches is proposed according to two dimensions. The first dimension distinguishes syntactic and semantic aspects. And the other is partitioned into functional and non-functional categories. Moreover, the functional mismatches can be further divided into signature and protocol subcategories, as shown in Table 1. Therefore, the problem space of service composition mismatches has been divided into six subcategories. Note that some kinds of mismatches are not included in the presented classification, e.g. domain-related [2] and technical mismatches [3], as these mismatches can not be conquered by means of service mediation. Besides, neither version nor configuration mismatches are included, since they can be decomposed to the presented subcategories. All of the subcategories will be respectively introduced in the following subsections. For clear representation, this paper adopts the irreplaceability perspective, which is based on the scenario the required interface can not be exactly replaced by the provided interface.

2.1. Syntactic-level signature mismatches

The signature of a service's interface refers to the profile of the interface specifying the structure of parameters and related data types as well as possible exceptions. Thereby, the syntactic-level signature mismatches are those occurring at the structure of a single method's signature. Here a meta-model of the method definitions of interfaces is presented and all possible kinds of syntactic-level signature mismatches can be enumerated based on the meta-model as follows:

A meta-model of the method definition:
Method (inDt_1 inPa_1, ..., inDt_n inPa_n,
outDt_1 outPa_1, ..., outDt_m outPa_m)
throws (exception_1, ..., exception_k)

(1) Mismatches of the structure of parameter list
- Extra parameters: The provided interface has extra parameters that the required interface does not need.
- Missing parameters: The provided interface has missing parameters that the required interface needs.
- Splitting parameters: One of parameters of the provided interface should be split into two or more parameters of the required interface.
- Merging parameters: Two or more parameters of the provided interface should be merged into one parameter of the required interface.
- Ordering of parameters: The sequence of parameter list of the provided interface should be reordered to match that of the required interface.

(2) Mismatches of the structure of data types
- Extra data fields: Some data type of the provided interface has extra data fields that the required interface does not need.
- Missing data fields: Some data type of the provided interface has missing data fields that the required interface needs.
- Splitting data fields: Some data field of a data type of the provided interface should be split into two or more data fields of a complex data type of the required interface.
- Merging data fields: Two or more data fields of a complex data type of the provided interface should be merged into one data field of a data type of the required interface.
- Ordering of data fields: The sequence of data fields of a complex data type of the provided interface should be reordered to match that of the required interface.

(3) Mismatches of the exception list
- Extra exceptions: The provided interface has extra exceptions that the required interface does not need.

- Missing exceptions: The provided interface has missing exceptions that the required interface needs.
- Typing of exception: Exceptions thrown by methods of the provided interface have a different type of that of the required interface.

2.2. Semantic-level signature mismatches

Semantic-level signature mismatches are conceptual mismatches referring to the signature of the provided and required interfaces. Note that naming and value range mismatches of parameters and date types are two well-known kinds of signature mismatches and put into the subcategories of semantic-level signature mismatches. As introduced in [4] [3], these mismatches can be distinguished as follows:

(1) Mismatches of synonyms

Two concepts, which characterize corresponding interface elements of a provided and required interface, are identical with respect to their definition, but have been used with different terms, particularly, including naming mismatches of methods, parameters and date types, etc.

(2) Mismatches of sub- and super-ordination

Two concepts, which characterize corresponding interface elements of a provided and required interface, are in a specialization or generalization relationship to each other.

(3) Mismatches of homonyms

Two concepts, which characterize corresponding interface elements of a provided and required interface, are named with the same term but have different definitions. Particularly, value range mismatches can be put into this subcategory. For instance, "age" is a parameter of a method of the required interface and has its value range between zero and 150 while the "age" parameter of the provided interface has no value constraints and can be negative.

(4) Mismatches of equipollence

Two concepts have the same extension. However, they have different definitions which only share some common aspects (e.g. customer vs. debtor).

2.3. Syntactic-level protocol mismatches

Protocol mismatches refer to mismatches occurring at the message exchanging sequences of the provided and required interfaces, which should be distinguished with semantic mismatches of interfaces' protocols and identified at the syntactic level. The existing researches have identified this kind of mismatches [5] [6]. However, few paper claims its identification is complete in any sense. To achieve a complete identification of syntactic-level protocol mismatches, this paper refers to four basic workflow constructs presented in [7] [8], namely sequence, parallel, exclusive choice and iteration. Other workflow patterns are supposed to be composed by using basic workflow constructs. Derived from basic workflow constructs and message exchanging sequences, six basic protocol mismatch patterns are identified in the following. Herein, the authors would like to point out the six basic patterns can be viewed as basic constructs of existing protocol mismatches, since protocol mismatches corresponding to basic workflow constructs can be straightly composed. For the sake of convenient illustration, NULL ACTIVITY is used to represent such activity that sends/receives no messages. Basic patterns of syntactic-level protocol mismatches are as follows:

(1) Mismatches of extra messages

The provided interface has some extra messages the required interface does not expect to send/receive.

Illustration: After receiving purchase order, the provided interface has to send receipt while the required interface expects no such action, as shown in Figure 1(a).

(2) Mismatches of missing messages

The provided interface does not have some messages the required interface expects to send/receive.

Illustration: After receiving purchase order, the provided interface does not send receipt while the required interface expects to send receipt, as shown in Figure 1(b).

(3) Mismatches of splitting messages

The provided interface has some messages the required interface expects to split to send/receive.

Illustration: After receiving purchase order, the provided interface sends the information of product availability and quote in a single message, while the required interface may send product availability ahead of product quote, as shown in Figure 1(c).

(4) Mismatches of merging messages

The provided interface has some messages the required interface expects to merge to send/receive.

Illustration: After receiving purchase order, the provided interface sends the information of product availability ahead of product quote, while the required interface expects to send product availability and quote in a single message, as shown in Figure 1(d).

(5) Mismatches of extra conditions

The provided interface has some extra conditions imposed on control flow of the protocol while the required interface expects no conditions.

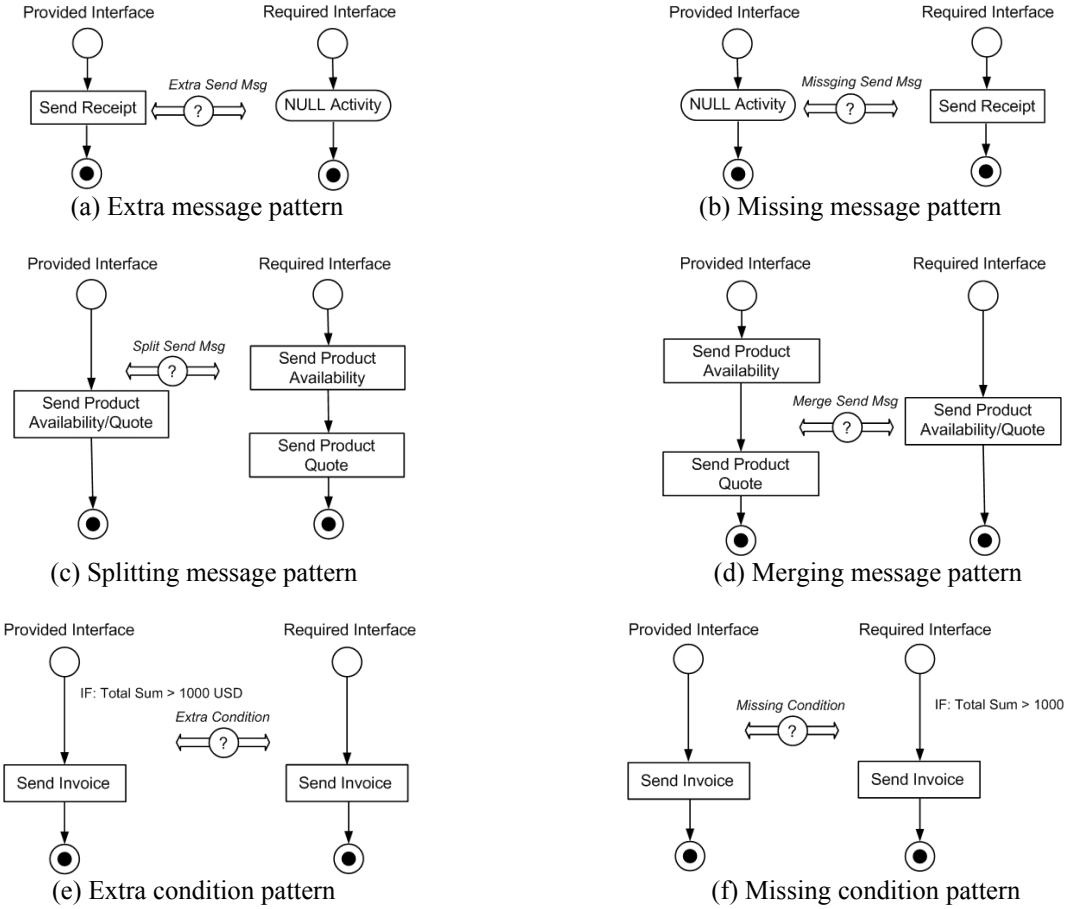
Illustration: After receiving payment of the purchase order, the provided interface sends an invoice

if the total sum is greater than 1000 USD. However, the required interface expects to directly send an invoice after payment, as shown in Figure 1(e).

(6) Mismatches of missing conditions

The provided interface has no conditions imposed on control flow of the protocol while the required interface expects to have some conditions to constrain its control flow.

Illustration: After receiving payment of the purchase order, the provided interface directly sends an invoice, while the required interface expects to do it under the condition that the total sum is greater than 1000 USD, as shown in Figure 1(f).



(a) Extra message pattern (b) Missing message pattern

(c) Splitting message pattern (d) Merging message pattern

(e) Extra condition pattern (f) Missing condition pattern

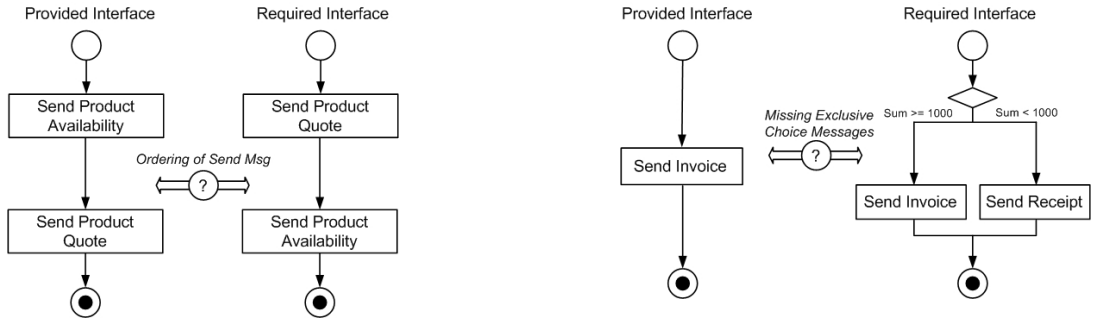
Figure 1. Basic protocol mismatch patterns

As mentioned above, syntactic-level protocol mismatches can be composed by using the basic mismatch patterns. Herein take two mismatches, namely *ordering of messages mismatch* and *missing exclusive choice mismatch*, for example, as shown in Figure 2(a) and Figure 2(b). The ordering of messages mismatch can be recognized as the composition of extra message pattern and missing message pattern. And the missing exclusive choice mismatch can be composed by the missing condition pattern and missing message pattern. The possible composition steps are illustrated, as shown in Figure 3 and Figure 4. Moreover, protocol mismatches derived from the repetition control, namely *Collapse* and *Burst* in [6], can be composed as well. Due to the length limitation, detailed procedures of composition are not presented.

2.4. Semantic-level protocol mismatches

Semantic-level protocol mismatches refer to conceptual mismatches of the behavioral processes of the provided and required interfaces. A significant situation classified in this subcategory is the different process declarations of the provided and required interfaces, which has not been identified in the existing works. For example, the declaration of the required interface' processes is based on Web services standards of BPEL and WSDL while that of the provided interface is specified with Interface

Description Language (IDL) or UML diagrams. Probably, it is required to develop mediators performing declaration transformations to conquer this kind of mismatches, which belongs to the future work of a systematic solution.



(a) Ordering of messages mismatch (b) Missing exclusive choice mismatch

Figure 2. Two other syntactic-level protocol mismatches

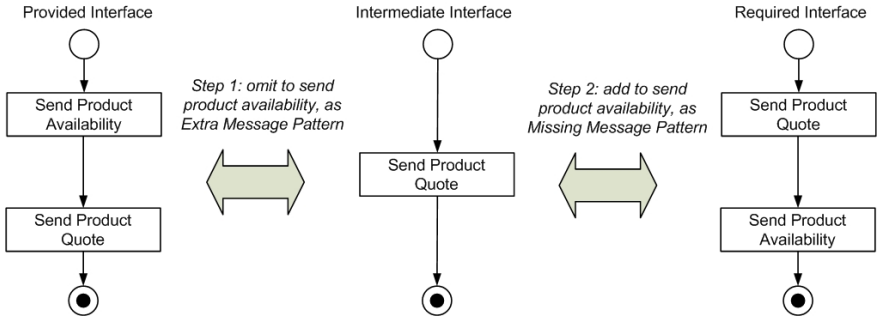


Figure 3. Composition steps of ordering of messages mismatch

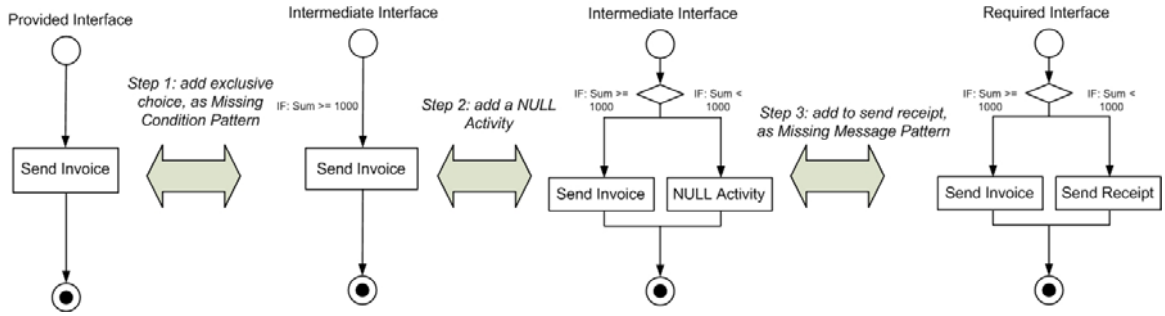


Figure 4. Composition steps of missing exclusive choice mismatch

2.5. Syntactic-level and semantic-level non-functional mismatches

Non-functional mismatches have attracted some attentions of academic researchers [3] [2], which can be divided into syntactic and semantic levels. On the one hand, syntactic-level non-functional mismatches are recognized as mismatches referring to the physical Quality of Service (QoS). And to identify all kinds of these mismatches, a QoS model is required to document quality attributes of services, such as QoS models introduced by Quality Modeling Language (QML) or ISO 9126 [9]. For instance, "reliability" is a well-known quality attribute and the required service may need a reliability of 95% while the provided service only 85%. On the other hand, semantic-level non-functional mismatches refer to inconsistent conceptual declarations of the quality attributes. Take "reliability" for example again. Although the reliability of required and provided services are both 95%, the reliability of the required service may have the declaration of continuity during the whole week, while that of the provided service means continuity from Monday to Friday excluding the weekends.

0-7695-2871-6/07 $25.00

3. Related work and comparisons

Recently, there have been a significant number of researches on service mediation and attempting to address various kinds of composition mismatches [3]. A checking mechanism on signature level mismatches has been proposed in [10] and several formal approaches have been developed to conquer protocol mismatches, such as automata [11], process algebra [12] and petri nets [13], etc. Among the current literatures, however, few of them are dedicated to the classification of composition mismatches and most of the existing classification approaches just broadly distinguish between signature, protocol, semantic and non-functional mismatches [14]. Although the taxonomy of service mismatches presented in [3] covers several categories of mismatches, it does not consider some categories should be classified from different dimensions. Moreover, the taxonomy does not sufficiently address protocol mismatches which are crucial and have to be dealt with. The work in [5] presents five protocol mismatch patterns, namely *Message Ordering*, *Extra Message*, *Missing Message*, *Message Split* and *Message Merge*. And two more protocol mismatches derived from the repetition control, namely *Collapse* and *Burst*, have been introduced in [6]. To the best of the authors' knowledge, however, few papers in the literature have addressed the composition and completeness of these protocol mismatches. And several protocol mismatches have firstly been identified in this paper, such as *Extra Condition*, *Missing Condition* and *Missing Exclusive Choice*. An important work addressing the classification issues is presented in [2] and has been introduced four classifications of composition mismatches. However, these classifications are either too broad to focus on or too complicated to conduct service mediation.

The main differences between this paper and others are: (1) a comprehensive and systematic classification of service composition mismatches, which covers most kinds of mismatches and becomes the foundation of developing a solution of service mediation; (2) six basic protocol mismatch patterns. Particularly, this paper addresses composition of protocol mismatches and points out these basic protocol mismatch patterns can be viewed as basic constructs of protocol mismatches, according to workflow patterns.

4. Conclusion and future work

The ultimate objective of the authors is to develop a systematic engineering solution for resolving composition mismatches and (semi-) automatically generating service mediators. As the foundation of the work, a comprehensive classification of most kinds of composition mismatches is presented, which is helpful to understand the problem space in deep. Another contribution of this paper lies in the identification of six basic protocol mismatch patterns which can be viewed as basic constructs of protocol mismatches. In the future, the authors would like to investigate the development of service mediators to address each kind of the identified mismatches, especially signature and protocol. And a systematic solution of service mediation is expected to be developed. Besides that, a prototype system will be constructed to validate the feasibility and efficiency of the solution.

5. Acknowledgements

This work was granted by National Natural Science Foundation of China (No. 60674080) and National High-Tech R&D (863) Plan of China (No. 2006AA04Z166 and No. 2006AA04Z157). Also, the work was supported by IBM-Tsinghua joint project "Mediator-aided Service Composition" co-sponsored by IBM China Research Lab and National CIMS Engineering Research Center, Tsinghua University, China.

6. References

[1] C. Canal, P. Poizat, and Gwen Salaun, "Adaptation of Component Behaviour using Synchronous Vectors", www.lami.univ-evry.fr/~poizat/documents/publications/RR-CPS05.pdf.

[2] S. Becker, S. Overhage, and R. Reussner, "Classifying Software Component Interoperability Errors to Support Component Adaption", Proc. of LNCS, Vol. 3054, Berlin, Heidelberg, Springer (2004) pp. 68-83.

[3] S. Becker, A. Brogi, and I. Gorton, et al., "Towards an Engineering Approach to Component Adaptation", Proc. of LNCS, Vol. 3938, Berlin, Heidelberg, Springer (2006) pp. 193-215.

[4] M. Paolucci, T. Kawamura, and T. Payne, et al., "Semantic Matchmaking of Web Services Capabilities", Proc. of 1st Intl. Conf. on Semantic Web, LNCS Vol. 2342, Berlin, Heidelberg, Springer (2002) pp. 333-347.

[5] B. Benatallah, F. Casati, and D. Grigori, et al., "Developing Adapters for Web Services Integration", Proc. of the 17th Intl. Conf. on Advanced Information System Engineering, CAiSE 2005, Porto, Portugal, Springer (2005) pp. 415-429.

[6] M. Dumas, M. Spork, and K. Wang, "Adapt or Perish: Algebra and Visual Notation for Service Interface Adaptation", Proc. of the 4th Intl. Conf. on Business Process Management, Springer (2006) pp. 65-80.

[7] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros, "Workflow Patterns", Journal of Distributed and Parallel Databases (2003), 14(1) pp. 5-51.

[8] W.M.P. van der Aalst, "The Application of Petri Nets to Workflow Management", Journal of Circuits, Systems and Computers (1998), 8(1) pp. 21-66.

[9] ISO/IEC: Software Engineering - Product Quality - Quality Model. ISO Standard 9126-1, International Organization for Standardization (2001).

[10] X. Xie, W. Zhang, "A Checking Mechanism of Software Component Adaptation", Proc. of the 5th Intl. Conf. on Grid and Cooperative Computing (GCC 2006) pp. 347-354.

[11] D. Yellin, R. Strom, "Protocol Specifications and Component Adaptors", ACM Transactions on Programming Languages and Systems 19 (1997) pp. 292–333

[12] A. Bracciali, A. Brogi, and C. Canal, "A Formal Approach to Component Adaptation", Journal of Systems and Software (2005) pp. 45–54.

[13] W. Tan, F. Rao, and Y. Fan, et al., "Compatibility Analysis and Mediation-aided Composition for BPEL Services", accepted by Intl. Conf. on Database Systems for Advanced Applications, DASFAA 2007.

[14] M. Hafedh, Y. Sherif, and A. Edward, "Reuse-Based Software Engineering: Techniques, Organization, and Controls", John Wiley, December 2001.