Handling Irreconcilable Mismatches in Web Services Mediation*

Xiaoqiang Qiao[1], Quan Z. Sheng[2], and Wei Chen[1]

[1] Institute of Software, Chinese Academy of Sciences
Beijing, 100190, China
{qxq,wchen}@otcaix.iscas.ac.cn
[2] School of Computer Science, the University of Adelaide
Adelaide, SA 5005, Australia
michael.sheng@adelaide.edu.au

Abstract. Service mediation provides an effective way to integrate a service requester and a service provider, by reconciling the mismatches between the two. The techniques to assess the mediation degrees of services, to analyze irreconcilable mismatches, and to provide resolutions for irreconcilable behavioral mismatches are therefore essential. To address these challenges, we introduce in this paper two quantifiable metrics, called *service mediatability* and *modification complexity*, to evaluate the feasibility and complexity of mediating a requester and a service. We also propose a pattern-based approach for analyzing service behaviors that cannot be automatically mediated. We further offer resolutions for each irreconcilable mismatch pattern, which help developers to adjust and improve the service behaviors to fulfill the interaction requirements.

1 Introduction

In order to interact seamlessly, a service requester and a Web service should be compatible both in *signature* and in *behavior* [3]. Service mediation is a feasible technique to deal with incompatible services by introducing extra components such as *service mediators* (or adaptors) [11]. Most existing approaches for Web service mediation only focus on how to synthesize service mediators semi-automatically or automatically in the case when services could be mediated. If there are irreconcilable mismatches, the services are simply considered as "not mediatable" and no further solution can be taken for mediation.

However, in practice, interactions among many services may not be fully mediated due to irreconcilable mismatches. Therefore, it is of great significance for analyzing and resolving irreconcilable mismatches between Web services. On the one hand, the irreconcilable information could be readily applied to measure i) the mediation degree of a given service and ii) the difficulty degree in amending the service request for a service mediation. Since there are usually multiple

* This work has been partially supported by the National High Technology Research and Development Program of China (863) under Grant No. 2012AA011204, the National Natural Science Foundation of China under Grant No. 61100065.

X. Franch et al. (Eds.): ICSOC 2014, LNCS 8831, pp. 445–452, 2014.

candidate services available for a specific request, such a measurement could be extremely useful for selecting the most suitable service with low cost. On the other hand, the irreconcilable information could also be used as a guide to modify the service request in order to mediate some selected Web services.

This paper focuses on services that could not be automatically mediated and advances the fundamental understanding on Web services mediation by proposing an approach for analyzing and assessing the irreconcilable behaviors of Web services. The main contributions of our work include: i) the concept of *mediatability* enabling a quantifiable measurement of mediation degrees between services, ii) a pattern-based method for analyzing service behaviors that cannot be mediated, iii) the corresponding solution for each irreconcilable pattern, and iv) a research prototype based on the proposed approach.

2 Mediation Degree Assessment for Service Interactions

Our proposed procedure for assessing the mediation degrees of services is illustrated in Fig. 1. First, the mediation model is constructed after defining the service and message mapping. Next, the mediation model is checked for verifying the existence of the mediator and calculating the mediatability of the services. Finally, if a service is mediatable, the corresponding mediator protocol will be automatically synthesized. Otherwise, a pattern-based analysis of the irreconcilable mismatches between the requester and the service will be conducted.

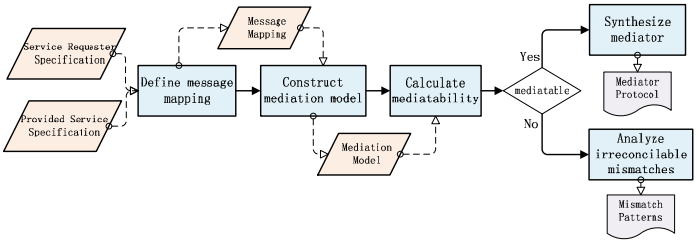


Fig. 1. The procedure of the proposed approach

2.1 Defining Service and Message Mapping

Definition 1. (Service). *A service is defined as a triple:* $\mathcal{S}=(\mathcal{M}_{in}, \mathcal{M}_{out}, \mathcal{P})$*:*

- $\mathcal{M}_{in}$ *is the finite set of messages that are received by service* $\mathcal{S}$*, and* $\mathcal{M}_{out}$ *is the finite set of messages that are sent by the service;*
- $\mathcal{P}$ *is the interaction protocol of service* $\mathcal{S}$ $\square$.

We adopt the process concept in Communicating Sequential Processes (CSP) [4] to model a service protocol. The language of CSP used in this paper is given in [8]. Message mapping indicates the message correlations between two services.

Definition 2. (Message Mapping). *Let interactive services be* $\mathcal{S}^A=(\mathcal{M}^A_{in}, \mathcal{M}^A_{out}, \mathcal{P}^A)$ *and* $\mathcal{S}^B=(\mathcal{M}^B_{in}, \mathcal{M}^B_{out}, \mathcal{P}^B)$*, the message mapping between them comprises two sets:* $Map_{<A,B>}$ *and* $Map_{<B,A>}$*.*

- $Map_{<A,B>}=\{synth_i(m_r, \mathcal{M}_D) | m_r \in \mathcal{M}^B_{in}, \mathcal{M}_D \subseteq \mathcal{M}^A_{out}, 1 \leq i \leq n\}$ *is a set of mapping rules from* $\mathcal{M}^A_{out}$ *to* $\mathcal{M}^B_{in}$*.* m_r *is a receiving message of service* $\mathcal{S}^B$ *and* $\mathcal{M}_D$ *is the set of sending messages of service* $\mathcal{S}^A$*.* $synth_i$ *is the mapping function to construct* m_r *from* $\mathcal{M}_D$*;*
- *Similarly,* $Map_{<B,A>}$ *is a set of mapping rules from* $\mathcal{M}^B_{out}$ *to* $\mathcal{M}^A_{in}$ □.

Based on the provided message mapping, we can apply behavior checking methods to determine whether irreconcilable mismatches exist. The mediation model specifies how two services exchange messages through a mediator, which could be automatically constructed based on message mapping.

Definition 3. (Mediation Model). *Let interactive service protocols be* $\mathcal{P}^A$ *and* $\mathcal{P}^B$*, the mediation model between them is:* $[Pipes||(\mathcal{P}^A_M||\mathcal{P}^B_M)]$*, where:*

- $Pipes = (||_{i \leq n} Pipe\_m_i)$*, here* n *is the number of the target messages defined in the message mapping. For each* $synth_i(m_r, \mathcal{M}_D)$*, there exists a corresponding message pipe* $Pipe\_m_i$ *and its behavior is described as:* $Pipe\_m_i = (||_{j \leq l} left?m_{dj}) \rightarrow synth_i \rightarrow right!m_r \rightarrow SKIP$ *where* $m_{dj} \in \mathcal{M}_D$*,* l *is the number of source messages that* m_r *depends on. A message pipe receives data with its* left *channel and writes the result message to the* right *channel.*
- $\mathcal{P}^A_M$ *and* $\mathcal{P}^B_M$ *are processes that in charge of reading messages from or writing messages into the corresponding pipes, which could be constructed from* $\mathcal{P}^A$ *and* $\mathcal{P}^B$ *respectively by replacing corresponding events based on the rules:*
  - $\forall !m \in \alpha\mathcal{P}^A$ *(or* $\alpha\mathcal{P}^B$*), if* m *is a source message in the message mapping,* $!m \Rightarrow (||_{i \leq n} Pipe\_m_i.left!m)$*. Here* n *is the number of target messages that depend on* m*. Otherwise,* $!m \Rightarrow WriteNull$*.* $WriteNull$ *is used to indicate that there is no specified reception for the sending message.*
  - $\forall ?m \in \alpha\mathcal{P}^A$ *(or* $\alpha\mathcal{P}^B$*), if* m *is a target message in the message mapping,* $?m \Rightarrow Pipe\_m.right?m$*. Otherwise,* $?m \Rightarrow ReadNull$*. Likewise, we use* $ReadNull$ *event to represent the required message could not be sent by the partner service.* □

We use the deadlock process concept in CSP to check the existence of mediator and locate the irreconcilable mismatches. To automatically perform the checking process, we further improve the algorithm in [9] to quantify the mediation degree of a service.

2.2 Calculating Mediatability

In order to check the mediation model for verifying the mediator existence and calculating the mediatability, we use algebraic laws of non-deterministic choice ($\sqcap$) to obtain *interaction paths*, which represent possible interactive processes between two services in a certain interaction with the aid of the mediator. Due to space constraints, the details of the algebraic laws are described in [8].

Definition 4. (Interaction Path). *Suppose the standard forms of non-deterministic choice of* $\mathcal{P}_M^A$ *and* $\mathcal{P}_M^B$ *are* $(p_1^A \sqcap p_2^A \sqcap ... \sqcap p_n^A)$ *and* $(p_1^B \sqcap p_2^B \sqcap ... \sqcap p_l^B)$ *respectively, the behavior of* $\mathcal{P}_M^A || P_M^B$ *is:* $(p_1^A || p_1^B) \sqcap (p_1^A || p_2^B) \sqcap ... \sqcap (p_1^A || p_l^B) \sqcap (p_2^A || p_1^B) \sqcap ... \sqcap (p_n^A || p_l^B)$. *Each non-deterministic branch sub-protocol of* $\mathcal{P}_M^A || P_M^B$, $(p_1^A || p_1^B)$, $(p_1^A || p_2^B)$,..., $(p_n^A || p_l^B)$, *is a interaction path between* $\mathcal{P}_M^A$ *and* $\mathcal{P}_M^B$. □

Algorithm 1 shows the procedure to check and record the deadlock events of each interaction path between the requester and the provided service. Function *move* is invoked alternately to traverse all events of the input sub-protocols (line 2). The return value of function *move* has four types. *NoMove* indicates no event is checked during this invocation, while *Moved* means some events have been checked in the invocation. *SKIP* indicates the checking is finished and *ReadNull* means a *ReadNull* is encountered.

Algorithm 1. Deadlock Event Checking

Input: a sub-protocol of $\mathcal{P}_M^{Requester}$: p_1, a sub-protocol of $\mathcal{P}_M^{Service}$: p_2
Output: the deadlock event set: *events*

```
1.  while (true) do
2.    result1:= move (p1), result2 := move (p2);
3.    if (result1 = ReadNull ∨ result2 = ReadNull)
4.      if (result1 = ReadNull) record (events, p1); end if
5.      if (result2 = ReadNull) record (events, p2); end if
6.    else if (result1 = NoMove ∧ result2 = NoMove)
7.      record (events, p2); record (events, p1);
8.    else if (result1 = NoMove ∧ result2 = SKIP)
9.      record (events, p1);
10.   else if (result2 = NoMove ∧ result1 = SKIP)
11      record (events, p2);
12.   else if (result1 = SKIP ∧ result2 = SKIP)
13.     return events;
14.   end if
15. end while
```

If either result_1 or result_2 is *ReadNull* (line 3), or both of them cannot move forward (return *NoMove*, line 6), the corresponding events can cause a deadlock and should be recorded (i.e., the function *record*). In order to check the remaining parts of p_1 and p_2, we assume the deadlock is resolved and continue the algorithm (line 2). It is noted that the checking is performed from the perspective of the requester. In the scenario when both p_1 (i.e., the requester) and p_2 (i.e., the service) return *NoMove* (line 6), the corresponding event in p_2 firstly will be resolved (line 7). If either result is *SKIP* and the other result is *NoMove* (line 8 and line 10), all of the remaining events in the corresponding protocol will be recorded. If both result_1 and result_2 are *SKIP* (line 12), the checking procedure is finished. Algorithm 2 shows the details on function *move*.

Based on the recording of the deadlock events, we can calculate the mediatability between the requester and the service. The mediatability of one interaction path is computed as follows:

$$MD_{path} = 1 - \left(\frac{\mathcal{N}_{deadlocks}}{\mathcal{N}_{total}}\right) \quad (1)$$

where $\mathcal{N}_{deadlocks}$ is the number of the recorded deadlock events and $\mathcal{N}_{total}$ is the number of all receiving events in the interaction path. If $\mathcal{N}_{total}$ is 0, $\left(\frac{\mathcal{N}_{deadlocks}}{\mathcal{N}_{total}}\right)$ should be 0. Clearly, the value of the mediatability of one interaction path lies in the range of 0 and 1.

Algorithm 2. Move

Input: a protocol to be checked: p
Output: the checking result: $result$

```
1.  if (isSequential(p))
2.    for each subSequentialProtocol p_i do
3.      result := move(p_i);
4.      if (result=SKIP)
5.        hasMoved := true;
6.      else if (result=ReadNull)
7.        return ReadNull;
8.      else if (hasMoved=true∨result=Moved)
9.        return Moved;
10.     else return NoMove;
11.     end if
12.   end for
13.   return SKIP;

14. else if (isParallel(p))
15.   for each subParallelProtocol p_i do
16.     result_i := move(p_i);
17.   end for
18.   if (all result_i = SKIP)
19.     return SKIP;
20.   else if (∃ result_i = ReadNull)
21.     return ReadNull;
22.   else if (∃ result_i = Moved)
23.     return Moved;
24.   end if
25.   return NoMove;
26. else if (isExternalChoice(p))
27.   for each subChoiceProtocol p_i do
28.     if (isChosen(p_i))
29.       return move(p_i);
30.     end if
31.     return NoMove;
32.   end for

33. else
34.   for each event a_i do
35.     if (isWriting(a_i))
36.       writePipe(a_i);
37.       hasMoved := true;
38.     else if (isReading(a_i))
39.       if (canRead(a_i))
40.         hasMoved := true;
41.       else if (hasMoved)
42.         return Moved;
43.       else return NoMove;
44.       end if
45.     else if (a_i = ReadNull)
46.       return ReadNull;
47.     end if
48.   end for
49.   return SKIP;
50. end if
```

The mediatability between the requester and the service is calculated using:

$$MD_{service} = (\sum_{i=1}^{n} MD^{i}_{path})/n \tag{2}$$

Here MD^{i}_{path} is the mediatability of $path_i$ in the mediation model and n is the number of the interaction paths. Larger values of the mediatability indicate fewer deadlock events and higher mediation degrees.

2.3 Analyzing Irreconcilable Mismatches

We present here a pattern-based method to further analyze the irreconcilable behaviors. A *mismatch pattern* refers to those mismatches that can be reused to identify the irreconcilable behaviors between services.

The mismatch patterns identified in this paper and their corresponding resolving method are presented in Table 1. The interactions between the requester and the service with these mismatches could not be achieved through automated mediation method, but only through manual efforts to modify the protocol and construct the mediator. It is noted that the cost on modifying the requester protocol may be very different. For example, patterns 2 and 4 need the requester to improve and offer more interactive messages or branches, the cost involved will be higher than that of patterns 1 and 5. Since mediatability only measures the quantity of the deadlock events that need to be modified, and cannot reflect the cost and difficulty of the modification, we introduce another metric, named *modification complexity*. The modification complexity of each atomic operation, valued between 0 and 1, is listed in Table 2.

Table 1. Irreconcilable mismatch patterns

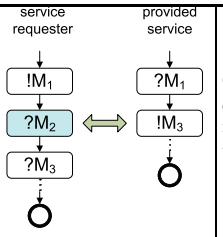
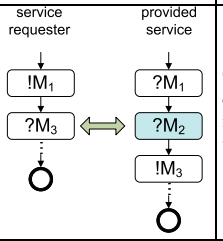
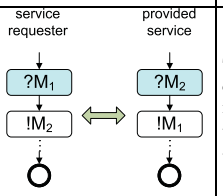
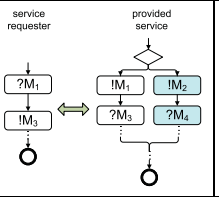
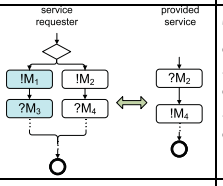
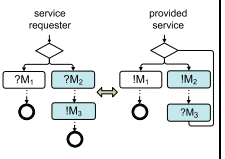
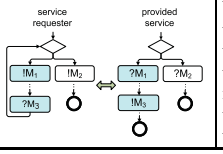
ID	Name	Description	Illustration	Checking Method	Resolving Method
1	Missing Requester Message	The service can not send a message that the requester expects to receive.		The deadlock events that are recorded when $result_1$ is *ReadNull*.	The requester deletes the corresponding event.
2	Missing Service Message	The requester can not send a message that the service expects to receive.		The deadlock events that are recorded when $result_2$ is *ReadNull*.	The requester adds the corresponding event to provide the required message.
3	Irreconcilable Ordering Mismatch	The message ordering mismatch that leads to a circular dependency.		The deadlock events that are recorded when $result_1$ and$result_2$ are *NoMove*.	The requester switches the ordering of the messages.
4	Missing Choice Branch in Requester	The entire choice branch in the service protocol has no counterpart.		The deadlock events belong to a choice branch of the service and the start event of the branch is *WriteNull*.	The requester provides the required choice branch.
5	Missing Choice Branch in Service	The entire choice branch in the requester protocol has no counterpart.		The deadlock events belong to a choice branch of the requester and the start event of the branch is *WriteNull*.	The requester deletes the required choice branch.
6	Missing Loop in Requester	A loop structure in the service protocol interacts with a non-loop structure in the requester protocol.		When p_2 ends with the loop flag while p_1 ends with *SKIP*, the receiving events in the loop structure would be recorded.	The requester changes the non-loop structure into the loop structure.
7	Missing Loop in Service	A loop structure in the requester protocol interacts with a non-loop structure in the service protocol.		When p_1 ends with the loop flag while p_2 ends with *SKIP*, the receiving events in the loop structure would be recorded.	The requester changes the loop structure into the non-loop structure.

Table 2. Modification Complexities of Atomic Operations

Operation	Patterns	Complexity
Add an event	Pattern 2 and 4	0.8
Delete an event	Pattern 1 and 5	0.4
Change the ordering of an event	Pattern 3	0.6
Change the execution times of an event	Pattern 6 and 7	0.6

Furthermore, the complexity of the control structure makes it difficult to modify the protocol, which should be also considered in calculating the overall modification complexity of the requester protocol. The formulas of calculating complexities for structural operators are shown in Table 3. The recursive structure involves a decision event and are executed multiple times. For the choice structures, the influence of the modification on other branches should be considered. In parallel structure, the execution of different branches should be synchronized. Therefore, these structures introduce extra difficulties to the protocol modification, and the corresponding weights are assigned to them.

Table 3. Complexity Formulas for Control Structures

Operator	Formula	Weight
$\rightarrow$	$MC_{a\rightarrow\mathcal{P}} = MC_a + MC_{\mathcal{P}}$	
$\Box$	$MC_{\mathcal{P}_1\Box\mathcal{P}_2\ldots\Box\mathcal{P}_n} = W_{\Box} * (MC_{\mathcal{P}_1} + MC_{\mathcal{P}_2} + \ldots + MC_{\mathcal{P}_n})$	$W_{\Box} = 1 + (n\text{-}1)/n$
$\sqcap$	$MC_{\mathcal{P}_1\sqcap\mathcal{P}_2\ldots\sqcap\mathcal{P}_n} = W_{\sqcap} * (MC_{\mathcal{P}_1} + MC_{\mathcal{P}_2} + \ldots + MC_{\mathcal{P}_n})$	$W_{\sqcap} = 1 + (n\text{-}1)/n$
;	$MC_{\mathcal{P}_1;\mathcal{P}_2\ldots;\mathcal{P}_n} = MC_{\mathcal{P}_1} + MC_{\mathcal{P}_2} + \ldots + MC_{\mathcal{P}_n}$	
$\|\|$	$MC_{\mathcal{P}_1\|\|\mathcal{P}_2\ldots\|\|\mathcal{P}_n} = W_{\|\|} * (MC_{\mathcal{P}_1} + MC_{\mathcal{P}_2} + \ldots + MC_{\mathcal{P}_n})$	$W_{\|\|} = 1.2$
$\mu\mathcal{X} \cdot \mathcal{F}(\mathcal{P};\mathcal{X})$	$MC_{\mu\mathcal{X}\cdot\mathcal{F}(\mathcal{P};\mathcal{X})} = W_{\mathcal{X}} * MC_{\mathcal{P}}$	$W_{\mathcal{X}} = 1.5$

3 Prototype Implementation and the Related Work

We have implemented a prototype system to validate the approach proposed in this paper. It provides editors to graphically specify the service protocol and edit the message mapping rules. It also provides facilities for the mediator existence checking. The interface of the prototype system is developed based on the Eclipse Plug-in technique and wrapped into an Eclipse Rich Client Platform (RCP) application. Due to space constraints, we will not give the details. Interested readers are referred to [8].

The works [1,6] analyze the possible types of mismatches between services and propose mediation patterns for developing mediators. [2,10,11] focus on automatic synthesis of mediator protocols. [5] adds semantic dependency relationship in the service description and presents a general process to derive concrete mediators from mediator specifications. However, none of these works analyzes the irreconcilable behaviors that lead to failure of mediated service interaction.

Nezhad et al. [7] provide some evidences that help to construct missing messages, and a very recent work by Zhou et al. [12] computes the number of irreconcilable interaction paths using a mechanism called *walk computation*. In this paper, we go a step further by focusing on quantitative assessment of mediation degree and modification complexity, pattern-based irreconcilable behavior

analysis, and mismatch resolution. Our proposed approach takes irreconcilable services into consideration when selecting Web services, thus increasing the range of candidate services. The resolutions for the irreconcilable patterns also reduce the complexity of manual adjustment for mediated service interactions.

4 Conclusion

In this paper, we advance the existing works on service mediation by proposing an approach to analyze and measure the irreconcilable behaviors for service mediation, including a quantifiable metric for measuring mediation degrees, a pattern-based method for mismatch analysis, a set of resolutions for irreconcilable patterns, and a further metric for measuring complexity and cost of modification in service mediation. Our proposed approach, particularly the two metrics, can also help developers in Web services selection. Our future work will extend the approach to support more complicated processes and investigate techniques developed by semantic Web initiatives to automate the service mediation process.

References

1. Benatallah, B., Casati, F., Grigori, D., Nezhad, H.R.M., Toumani, F.: Developing Adapters for Web Services Integration. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 415–429. Springer, Heidelberg (2005)
2. Canal, C., Poizat, P., Salaün, G.: Model-Based Adaptation of Behavioral Mismatching Components. IEEE Trans. Softw. Eng. 34(4), 546–563 (2008)
3. Dumas, M., Benatallah, B., Nezhad, H.R.M.: Web Service Protocols: Compatibility and Adaptation. IEEE Data Engineering Bulletin 31(1), 40–44 (2008)
4. Hoare, C.: Communicating Sequential Processes. Prentice-Hall (1985)
5. Kuang, L., Deng, S., Wu, J., Li, Y.: Towards Adaptation of Service Interface Semantics. In: Proc. of the 2009 IEEE Intl. Conf. on Web Services, ICWS 2009 (2009)
6. Li, X., Fan, Y., Madnick, S., Sheng, Q.Z.: A Pattern-based Approach to Protocol Mediation for Web Services Composition. Info. and Soft. Tech. 52(3), 304–323 (2010)
7. Nezhad, H., et al.: Semi-Automated adaptation of service interactions. In: Proc. of the 16th Intl. Conf. on World Wide Web, WWW 2007(2007)
8. Qiao, X., Sheng, Q.Z., Chen, W.: Handling irreconcilable mismatches in web services mediation. Tech. Rep. TCSE-TR-20140501, `http://otc.iscas.ac.cn/cms/UploadFile/20140731050648880/`
9. Qiao, X., Wei, J.: Implementing Service Collaboration Based on Decentralized Mediation. In: Proc. of the 11th Intl. Conf. on Quality Software, QSIC 2011 (2011)
10. Tan, W., Fan, Y., Zhou, M., Zhou, M.: A Petri Net-Based Method for Compatibility Analysis and Composition of Web Services in Business Process Execution Language. IEEE Trans. Autom. Sci. Eng. 6(1), 94–106 (2009)
11. Yellin, D.M., Strom, R.E.: Protocol Specifications and Component Adaptors. ACM Transactions on Programming Languages And Systems 19(2), 292–333 (1997)
12. Zhou, Z., et al.: Assessment of Service Protocol Adaptability Based on Novel Walk Computation. IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Humans 42(5), 1109–1140 (2012)