Process Mediation Based on Triple Space Computing

Zhangbing Zhou [1], Brahmananda Sapkota [1], Emilia Cimpian [2],
Doug Foxvog [1], Laurentiu Vasiliu [1], Manfred Hauswirth [1] and Peng Yu [3]

[1] DERI, National University of Ireland at Galway, Ireland
{zhangbing.zhou, brahmananda.sapkota, doug.foxvog,
laurentiu.vasiliu, manfred.hauswirth}@deri.org
[2] STI International, University of Innsbruck, Austria
emilia.cimpian@deri.org
[3] College of Computer Science and Technology, Jilin University, China
yupeng79@gmail.com

Abstract. Web services are inherently heterogeneous at both data and behavioral levels because of the nature of the Web, which is the main obstacle to the usability of Web services. The heterogeneity at a behavioral level is generally addressed by process mediation, in which the message flow is adjusted to suit the behavior of Web services involved in a given interaction. In this paper, we present a novel approach for process mediation, and propose an architectural for process mediation based on Triple Space Computing to solve resolvable message sequence mismatches. These resolvable mismatches can be classified into five classes for unveiling their essence. This work provides a basis for the generalization of mismatches themselves, as well as a potentially uniform solution to address these mismatches.

1 Introduction

Web services act as computational entities and the main pillar for Service-Oriented Architecture, and aim at supporting interoperable machine-to-machine interactions over the Web. Because of inherent *autonomy* and *heterogeneity*, Web services are heterogeneous at both data and behavioral levels. In general, the messages are often different in *format* and *granularity*, and public processes [9] are often diverse in activities and messages in terms of *form* and *sequence*. Data heterogeneities can be mitigated with the help of data mediation [1], while process mediation [12] aims at aligning different interaction patterns by adjusting bi-directional flows of messages. Process mediation is very valuable for complex service interactions in which several Web services may involve. However, process mediation is optional for RPC-style service interactions, where there is only a single request-response message exchange.

Based on the message exchange patterns specified for a Web service, called as the public process, process mediation aims at resolving message sequence mismatches for: (1) s*ervice discovery and selection*: to ensure that discovered and/or selected Web services are behaviorally compatible [15] with a given goal. A functional aspect is currently the focus for service discovery and selection. Process mediation would be used for identifying whether a goal and the services are compatible from a behavioral aspect, (2) *service composition*: process mediation would guarantee that there are only resolvable behavioral mismatches [12] among a goal and Web services, and (3) *service execution*: process mediation would instruct the exchange of messages among Web services, and thus smooth the interaction.

In this paper, we propose a novel approach of process mediation for dealing with message sequence mismatches. We apply our method to WSMO[1] based Web services. However, our method is general and can be applied to other semantic Web services (SWSs) models like OWL-S[2] at ease. A public process can be described by these SWSs conceptual models. In addition, we propose an architecture for process mediation based on Triple Space Computing (TSC) [8]. Potential solutions are presented for resolvable mismatches. Furthermore, we classify these mismatches into five classes. The main contributions of this paper are four-fold: (1) a novel approach of process mediation for dealing with behavioral mismatches, (2) an architecture for process mediation based on TSC, (3) potential solutions for resolvable message sequence mismatches, and (4) five classes for these resolvable mismatches.

The rest of the paper is organized as follows: In Section 2, we give an introduction to WSMO and TSC. In Section 3, we propose our TSC-based architecture for process mediation. In Section 4, we present potential solutions for resolvable message sequence mismatches and categorize them into five classes. In Section 5, we discuss related works. In Section 6, we conclude this paper and indicate our future work.

2 Background

Due to the space limitation, we give a brief introduction to WSMO in Section 2.1 and Triple Space Computing in Section 2.2.

2.1 WSMO

WSMO is one of the major SWSs conceptual models initiated by the Web Service Modeling Ontology working group[3] of the ESSI cluster[4]. WSMO defines four major components: *ontology*, *Web services*, *goal*, and *mediator*, following the framework proposed in WSMF [9]. Web services and goals have a common component: *an interface*, which specifies how their functionality can be achieved though a two-fold view of operational competence: *choreography* [10] and *orchestration* [14].

Choreography describes the behavioral interface of Web services by which a client can consume its functionality [3]. This means that it presents an interface from a user's point of view. A user can be a person, an application, or another Web service.

Orchestration defines the behavioral interface of a Web service for achieving its functionality by aggregating other Web services [18]. An orchestration could be

[1] http://www.wsmo.org/TR/d2/v1.3/.
[2] http://www.ai.sri.com/daml/services/owl-s/1.2/overview/.
[3] http://www.wsmo.org/.
[4] http://www.essi-cluster.org/.

regarded as a "*composition*" of several "*sub-goals*". Each "*sub-goal*", acting as a user, consumes another Web service through its choreography. However, an orchestration is optional if the functionality of a Web service could be achieved completely by its choreography.

2.2 Triple Space Computing

TSC is a persistent communication and coordination paradigm for application and service integration on the Web [21]. It is based on the convergence of Semantic Web [22] and tuple-space computing [7] technologies. TSC acts as a *globally accessible*, *Web-scaled* and *space-like* middleware to enable so-called Web paradigm: *information is persistently written to a globally shared space where other processes can smoothly access it without starting a cascade of message exchanges* [8]. Triple Spaces introduce an infrastructure that enables machines to use an equally powerful communication medium in the same way as the humans use the Web [21]. The main advantages that TSC brings are four-fold: (1) *time autonomy*: the only time dependency is that RDF triples must be written before they can be read, (2) *location autonomy*: storage location provided by Triple Space is independent to that of the provider or the requester, (3) *reference autonomy*: the provider and the requester do not need to know each other and there is no explicit communication channel between them. They exchange information by writing and reading RDF triples to and from a Triple Space, and (4) *data schema autonomy*: data written to and read from a Triple Space would follow TSC data schema, which follows RDF specification. This makes the provider and the requester independent of their internal data schemas.

Besides the functionality of space-based computing, TSC offers more features such as transaction support, distribution, and query using RDF format etc. For more information about TSC, the reader can visit the web page of TripCOM Project[5].

3 Process Mediation Integration with TSC

In this Section, we propose an approach of process mediation addressing behavioral mismatches for WSMO-based Web services, and present a TSC-based architecture for process mediation.

3.1 Process Mediation Framework

Process mediation bridges potential behavioral mismatches between a user and a Web service chorography, or between a "*sub-goal*" of a Web service orchestration and the choreography of another Web service. Data mediation is necessary to support process mediation if the goal and the services are represented using different ontologies. In

[5] http://tripcom.org/deliverables.php.

order to support this requirement, we propose a framework for process mediation as shown in Figure 1. Process mediation would concentrate on behavioral compatibility [5] between two partners since the majority of interactions are often related to two partners, and an interaction involving multiple partners can often be decomposed into several pair-wise interactions.

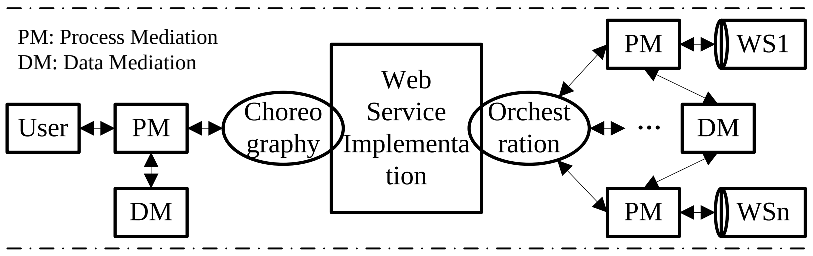

Fig. 1. Process Mediation Framework Based on WSMO

3.2 Architecture

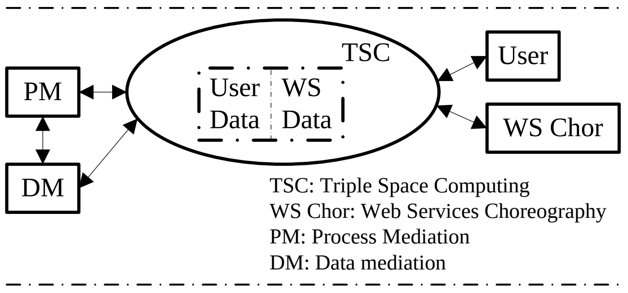

Fig. 2. Integrating Process Mediation with Triple Space Computing

TSC-based process mediation architecture is presented in Figure 2. Data mediation addresses potential data heterogeneity problems if a user and a Web service are described by different ontologies. TSC brings machine-to-machine Web service interaction to Web scale, and acts as a communication middleware between a user and a Web service (actually the choreography of Web services). There is a virtual data space within TSC for a given interaction, which includes two sub-spaces: one for the user and another for the Web service. Data would follow TSC data schema. The user and the Web service send requests to and retrieve responses from TSC. They do not communicate directly. Based on data stored in TSC, process mediation could handle possible behavioral mismatches between the user and the Web service.

Typically, Web service interactions are based on the message exchange paradigm, and often need to establish *synchronous* and *stateful* conversations. Therefore, they require a strong coupling in terms of reference and time [16]. However, these strong couplings do not exist in our architecture. The major advantages that TSC brings for service interactions are:

Backend storage: TSC provides a global, Web-scale space middleware for service interaction as well as process mediation. All data to be exchanged in an interaction is available in a shared virtual data space.

Asynchrony: TSC acts as a message broker between Web services. Therefore, Web services do not need to know each other explicitly, and communication channels between Web services are unnecessary [6].

Without loss of generality, we assume that the messages exchanged between Web services in a given interaction are asynchronous because synchronous can always be translated into a pair of asynchronous request-response calls [17]. There are three kinds of possible response for a request: *data message* to provide the information requested, *acknowledgement (ACK)* to indicate that a message sent was received by the receiver and the message is syntactically correct. An ACK does not suggest that the message is semantically valid. Or *negative acknowledgement (NAK)* to indicate that the data received is invalid in syntax.

State archiving for Web service interactions: TSC allows the storing of the *history* of the interactions, which is a flow of messages to and from TSC. This provides three main advantages: (1) this enables the monitoring of communicating applications, therefore helping in reuse of available Web services [6], (2) the archived messages represent the observed behavior of Web services. Therefore, we can check whether this observed behavior is consistent with the modeled behavior for a given Web service, and (3) the archived messages represent the locale of a given interaction. If the interaction fails for some reason, the interaction could resume from the point of failure with a possible manual adjustment, but does not need to start again from scratch. This is critical for long-running or non-repeatable interactions.

Semantic autonomy: TSC data model acts as intermediate model for the partners involving in a given interaction. They do not need to agree on a common data representation. Therefore, data mediation is not mandatory.

4 Behavioral Mismatches and Potential Solutions

A Web service interaction, especially that of complex Web services, often requires a flow of messages exchanged among the partner(s), and needs to maintain a state internally [2]. There may be some mismatches between the exchanged messages if the interaction is not perfect-match. Perfect-match means that the partners have exactly the same pattern to realize their public processes, and thus the messages sent by one partner are exactly the same in terms of *order* and *granularity* as requested by its corresponding partner [12]. In this case, only data mediation would be used to address possible data heterogeneity problems.

In this context, the mismatches could be classified into two categories: irresolvable message sequence mismatches (Section 4.1) and resolvable message sequence mismatches [12] (Section 4.2 and 4.3).

4.1 Irresolvable Message Sequence Mismatches

Irresolvable message sequence mismatches means that these mismatches cannot be handled automatically. As stated in [12], the following two scenarios are presented as irresolvable message sequence mismatches: (1) one partner wants to receive a message that the other does not want to send, thus the interaction fails because of lacking a required message, and (2) one partner expects an ACK for a certain message, but the other does not want to receive this message. Process mediation cannot generate such an ACK. Otherwise the entire communication is changed.

These types of mismatches, as stated in [9], could only be resolved by either of the following two solutions: (1) to change the interface definition for the goal and/or the Web service to avoid mismatches, or (2) to operate manually (such as skipping the activity that causes a fault) to bypass the failure.

However, Web service interface is often not allowed to be changed arbitrarily, and it is inappropriate to allow unexpected human interventions during service executions. This indicates that it is inappropriate that irresolvable mismatches exist in the public processes, and irresolvable mismatches are out of the scope process mediation.

4.2 Resolvable Mismatches and Potential Solutions

The left side of Figure 3 presents five scenarios for resolvable message sequence mismatches taken from [12]. These five scenarios are atomic mismatches, and complex mismatches can be built recursively by applying these five atomic ones. The right side presents corresponding data transformation in TSC data space. Process mediation has a *priori* knowledge of ontologies used by the business partners. Below we introduce these mismatch scenarios and their potential solutions:

Scenario A: suppose that BP1 sends messages “a” and “b” to BP2, but only “b” is expected by BP2. Process mediation should retain and store “a” for possible later use.
Potential Solution: “a” and “b” are sent to TSC and stored in the sub-space for BP1. Based on BP2’s ontology, process mediation knows that only “b” is expected, process mediation adds or updates data instance for “b” in the sub-space of BP2.

Scenario B: suppose that BP1 sends messages “a” and then “b” to BP2, while BP2 expects to receive “b” and then “a”. Process mediation reverses the ordering of these two messages.
Potential Solution: “a” and “b” are sent to TSC and stored in the sub-space of BP1. Based on the ontology of BP2, process mediation knows that both “a” and “b” are expected by BP2, and process mediation adds or updates data instances for both “a” and “b” in the sub-space of BP2.

Scenario C: suppose that BP1 sends both “a” and “b” in a single message to BP2, while BP2 expects to receive “a” and “b” separately. According to BP2’s request, process mediation should split this single message from BP1.
Potential Solution: There are two kinds of possible reasons for this mismatch:
- BP1 and BP2 use different ontologies: (“a” + “b”) is modeled by one concept in the ontology of BP1, while “a” and “b” are modeled by different concepts in the ontology of BP2.

- The ontologies of BP1 and BP2 are the same for “a” and “b”: one concept for “a” and another concept for “b”. But the messages are coded in different granularities. The granularity of a message indicates the number of the concepts implied by the data instances in this message. The *less* the number of concepts in a message, the *finer* the message is. In Scenario C, the messages of BP2 are finer than that of BP1, because the message of BP1: (“a” + “b”) implies two concepts, while the messages of BP2: “a” and “b” imply only one concept.

Data mediation knows the mapping for BP1 (“a” + “b”) to BP2 (“a”) and BP2 (“b”). Process mediation adds or updates data instances for “a” and “b” in the sub-space of BP2.

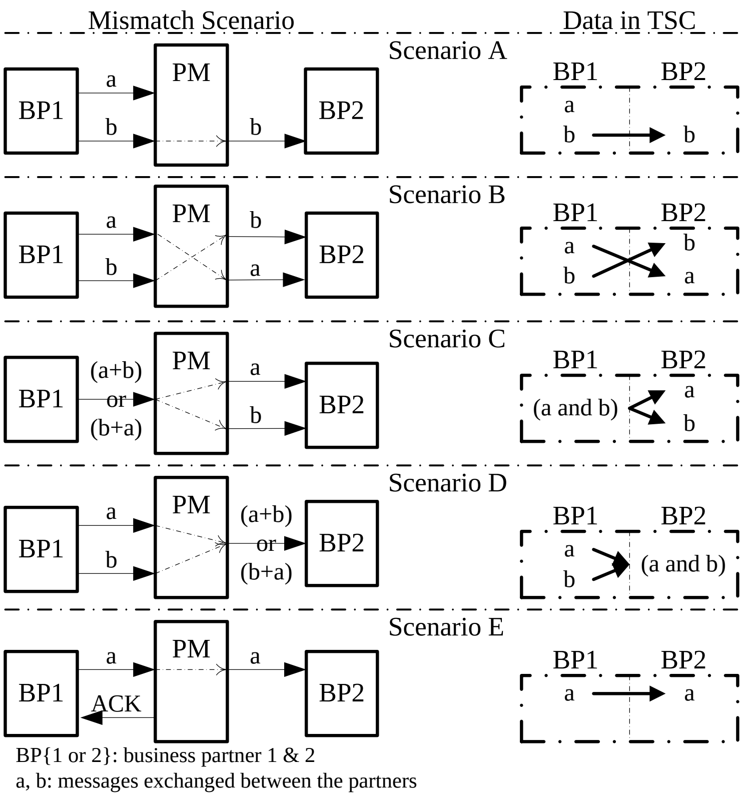

Fig. 3. Resolvable Message Sequence Mismatches

Scenario D: suppose that BP1 sends two messages “a” and “b” to BP2, while BP2 expects to receive them in one single message. Process mediation should combine two messages from BP1 into one message for BP2.
Potential Solution: This is the reverse case of Scenario C, and could be solved in the similar way as presented in Scenario C.

Scenario E: suppose that BP1 sends a message “a” to BP2 and expects an ACK for “a”. If “a” is expected by BP2, and BP2 is not willing to send an ACK back, process mediation should generate this ACK and send it back to BP1.

Potential Solution: “a” is sent to TSC and stored in the sub-space for BP1. Process mediation knows that “a” is expected by BP2, so it updates the sub-space of BP2 for “a” and generate an ACK for BP1. If “a” is either wrong in format or invalid in syntax, a NAK is returned. Process mediation has the knowledge of message formats for both BP1 and BP2.

4.3 Discussion

In this section, we analyze these resolvable mismatches shown in Figure 3. They can be categorized into five classes:

Class One: Extraneous data. Scenario A in Figure 3 is a good example of this class: one partner provides more data than what its partner wants to receive. However, this is only applicable to asynchronous communication. If the interaction is synchronous, this falls into irresolvable mismatches. An example is illustrated in Figure 4, which is synchronous counterpart of Scenario A:

BP1 expects a response for “a”. “a” is not expected by BP2, which means that BP2 has not a *priori* knowledge about “a”, and is uncertain whether “a” is necessary or not. If process mediation generates an ACK (or NAK) to BP1 for “a”, ACK (or NAK) indicates that “a” is expected by BP2 (while NAK indicates that “a” is expected by BP2, but “a” is not correct in terms of *format* or *syntax*). Obviously, ACK (or NAK) conveys a misleading indication to BP1 and changes the interaction. Therefore, the process mediation could not generate and send an ACK (or NAK) back to BP1, and this is a scenario of irresolvable mismatches.

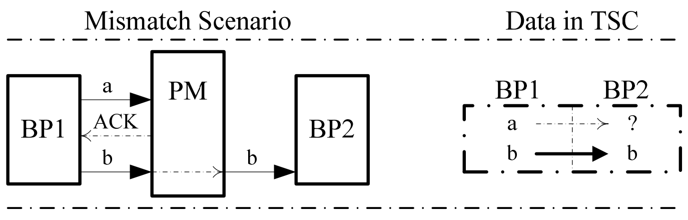

Fig. 4. Synchronous Communication for Extraneous Data

Class Two: Interaction with synchronous and asynchronous communication. Scenario E in Figure 3 falls into this class, which means that one partner is synchronous while the other is asynchronous. Based on the discussion in Class One, it is clear that Class Two is resolvable only if the data sent is expected by the receiver.

Class Three: Data heterogeneity. Scenarios C and D in Figure 3 fall into this class, which means that the partners use different ontologies, and some mismatches exist between these ontologies. Data mediation aims at supporting process mediation for solving these data heterogeneity problems.

Class Four: Message granularity heterogeneity. Scenarios C and D in Figure 3 belong to this class. This means that the ontologies used by the partners are similar, but the messages of different partners are of different granularity. Process mediation aims at solving this message granularity heterogeneity by combining or splitting the messages.

Class Five: Unnecessary message sequence dependencies. Scenarios B and D in Figure 3 fall into this class. Message sequence dependency means the ordering of the messages. The necessary message sequence dependency means that message ordering should be held during execution phases. We further explain this by Scenario B in Figure 3. Firstly, we rename message “a” sending as MsgA, while message “b” sending as MsgB. The purpose of messages renaming is to indicate that this concept applies to both incoming and outgoing messages. Necessary message sequence dependency of MsgA and MsgB indicates that: MsgA should be sent or received before MsgB, while MsgB should not be sent or received before MsgA.

Based on Scenario B, Figure 5 gives an example for unnecessary message sequence dependency, and shows that an unnecessary dependency can be removed. In scenario B, BP1 is willing to send “a” and then “b”. However, BP1 could reverse the sequence by sending “b” and then “a”. The changed sequence has no impact to the behavior of BP1 because “b” does not depend on “a”. It is the same for messages “b” and “a” in BP2.

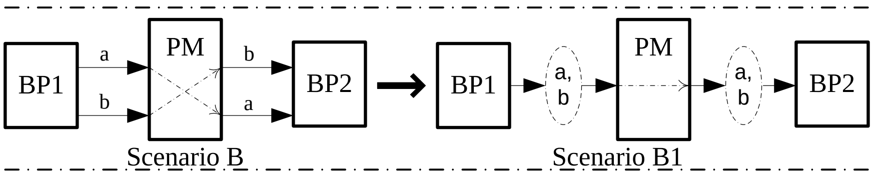

Fig. 5. Unnecessary Message Sequence Dependency

Based on this observation, we remove the sequence between “a” and “b” as presented in Scenario B1. The behavior specified by Scenario B1 is the same as that of Scenario B, while the message sequence mismatch in Scenario B disappears in Scenario B1. Scenario B1 is actually an example of perfect-match.

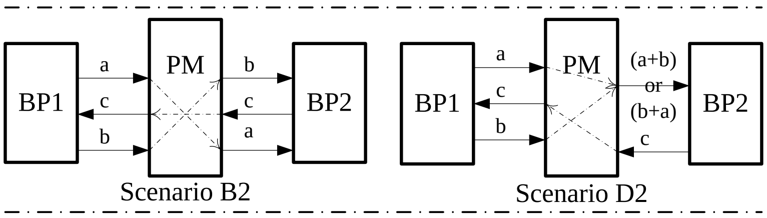

Fig. 6. Necessary Message Sequence Dependency

Figure 6 presents examples for necessary message sequence dependency in Scenario B2 and D2, which are based on Scenario B and D in Figure 3 with some change. These mismatches are irresolvable.

Scenario B2: BP1 is willing to send “a” then wait for “c”, and after that send “b”. “c” may be the response of “a”, therefore the content of “c” may depend on the content of "a". The same is for "c" and "b". Consequently, the sequence of "a", "c" and "b" is necessary and cannot be changed. Similar for BP2 that the sequences among "b", "c" and "a" are necessary and need to preserve. A failure occurs during this interaction because BP1 waits for "c" while BP2 waits for "b". Process mediation cannot generate "c" for BP1, as well as "b" for BP2.

Scenario D2: BP1 wants to send "a" then wait for "c", after that send "b". Based on the analysis of Scenario B2, the sequences among "a", "c" and "b" are necessary and cannot be changed. The same for BP2 that the sequence between ("a+b") or ("b+a") and "c" is necessary and should be preserved. The interaction between BP1 and BP2 fails because BP1 waits for "c" while BP2 waits for ("a+b") or ("b+a").

5 Related Works

The requirement of process mediation for supporting complex Web service interactions has been widely accepted as an important research topic. In [11], the authors present the scope of process mediation, list resolvable message sequence mismatches, propose an approach to integrate process mediation as a component into WSMX [19], and specify the interaction mode of process mediation with other components. In addition, they argue that process mediation would be used to support service invocation [13]. This work has been used in DIP project[6]. This work is a starting point of process mediation in WSMX. This paper benefits much from this work, especially message sequence mismatches in Section 4. However, the proposed approach is simple and suits the simplest workflow pattern: *sequence*, but would fail for complex ones such as *choice* or *loop* [20]. The work does not mention the importance of process mediation for complex service interactions, in which process mediation needs to support service discovery and selection to guarantee that the goal and the services are behaviorally compatible.

An adapter-based approach is proposed in [4] intending to semi-automatically resolve Web service differences at interface and business protocol levels. Possible differences between Web services are identified and captured by *mismatch patterns*. A pattern includes a business logic template, and can be used as a type of mismatch addressed by an adapter. However, *mismatch patterns* at the interface level are actually data heterogeneity problems covered by data mediation. *Mismatch patterns* at the business protocol level are the same as our resolvable mismatches listed in Figure 3. This work aims to formalize Web service protocols and interface/protocol *mismatch patterns*, and thus to provide a high-level framework as well as a uniform mechanism to address these mismatches.

In [23], the authors present the purpose of a process mediator within WSMX, which is a message broker among the partners. Process mediator needs to decide which data belongs to which partner(s) based on choreography and ontology of the partner(s). This work extends process mediation to multi-lateral interactions, and focuses on message forwarding among the partners. However, this data distribution

[6] http://dip.semanticweb.org/deliverables.html.

among the partners is actually, only a part of task that should be addressed by process mediation.

6 Conclusion and Future Works

Process mediation is a complex task and important for complex service interactions in which behavioral heterogeneity problems may exist in public processes. We argue that process mediation would aim at pair-wise interactions only. We propose a process mediation architecture based on TSC, and present potential solutions for resolvable message sequence mismatches. In addition, we categorize these resolvable mismatch scenarios into five classes. This analysis generalizes the resolvable message sequence mismatches, provides the basis for checking Web service compatibility from the behavioral aspect, and offers an opportunity to have a uniform solution to address these mismatches.

Process mediation in the context of SWSs is still in its infancy [11]. The related work is based on the exchanged messages, which represent a part of service behavior. It is a common sense that two public processes, which are locally compatible, do not necessarily mean that they are globally compatible. In the future, process mediation needs to consider compatible [5] on the public process level, and to check whether the mismatches are resolvable. Transitional support is another direction for ensuring the integrity of the interaction and the recovery in case of failure. Also, we aim to implement this proposal for evaluating it against the real data sets taken from the real-life use cases.

Acknowledgments. The work presented in this paper was supported (in part) by the EU funded TripCom Specific Targeted Research Project under Grant No. FP6-027324, and (in part) by the Lion project supported by Science Foundation Ireland under Grant No. SFI/02/CE1/I131.

References

1. Mocan, A., Cimpian, E. and de Bruijn, J.: D13.3v0.3 WSMX Data Mediation WSMX Working Draft 11. Available at http://www.wsmo.org/TR/d13/d13.3/v0.3/. (2005).
2. Wombacher, A.: Decentralized Consistency Checking in Cross-organizational Workflows E-Commerce Technology. In Proc. of the 8th IEEE International Conference on and Enterprise Computing, E-Commerce, and E-Services (2006).
3. Norton, B., Pedrinaci, C., Lemcke, J., Kleiner, M., Henocque, L. and Vulcu, G.: DIP Delivery: D3.9 An ontology for web services choreography and orchestration V3. Available at http://dip.semanticweb.org/deliverables.html. (2006).
4. Benatallah, B., Casati, F., Grigori, D., Nezhad, H.R.M. and Toumani, F.: Developing Adapters for Web Services Integration. In Proc. of Int'l Conf. Advanced Information System Eng. (CAISE '05), Porto, Portugal (2005).
5. Benatallah, B., Casati, F. and Toumani, F.: Representing, analysing and managing web service protocols. Data & Knowledge Engineering 58: (2006), 327-357.

6. Sapkota, B., Kilgarriff, E. and Bussler, C.: Role of Triple Space Computing in Semantic Web Services. In Proc. of the 8th Asia Pacific Web Conf (APWEB '06), (2006).
7. Gelernter, D.: Generative Communication in Linda. ACM Transactions on Programming Languages and Systems 7: (1985), 80-112.
8. Fensel, D.: Triple-space computing: Semantic Web Services based on persistent publication of information. In Proc. of IFIP Int'l Conf. on Intelligence in Communication Systems (2004).
9. Fensel, D. and Bussler, C.: The Web Service Modeling Framework WSMF Electronic Commerce Research and Applications. Page (2002), 113-137.
10. Roman, D., Scicluna, J., Nitzsche, J., Fensel, D., Polleres, A., de Bruijn, J. and Heymans, S.: D14v0.4. Ontology-based Choreography. WSMO Working Draft 15. Available at http://www.wsmo.org/TR/d14/v0.4/. (2007).
11. Cimpian, E. and Mocan, A.: WSMX Process Mediation Based on Choreographies In Proc. of the 1st International Workshop on Web Service Choreography and Orchestration for Business Process Management at the BPM 2005, Nancy, France. (2005).
12. Cimpian, E., Mocan, A. and Scicluna, J.: D13.7 v0.2 Process Mediation in WSMX WSMX Working Draft. Available at http://www.wsmo.org/TR/d13/d13.7/v0.2/. (2005).
13. Cimpian, E., Mocan, A. and Stollberg, M.: Mediation Enabled Semantic Web Services Usage. In Proc. of the 1st Asian Semantic Web Conference (ASWC '06), (2006).
14. Haas, H. and Brown, A.: Web Services Glossary W3C Working Group, Available at http://www.w3.org/TR/ws-gloss/. (2004).
15. Bordeaux, L., Salaün, G., Berardi, D. and Mecella, M.: When are Two Web Services Compatible? In Technologies for E-Services, 5th International Workshop (TES '04). Springer Berlin / Heidelberg, Toronto, Canada. (2004).
16. Nixon, L.J.B., Bontas, E.P. and Scicluna, J.: D2.4.8.1: Technical and ontological infrastructure for Triple Space Computing. Delivery of Knowledge Web project. Available at: http://knowledgeweb.semanticweb.org/semanticportal/sewView/frames.jsp. (2006).
17. Pistore, M., Traverso, P. and Bertoli, P.: Automated Composition of Web Services by Planning in Asynchronous Domains. In Proc. of the 15th Interational Conferece on Automated Planning and Scheduling (ICAPS '05), Monterey, California. (2005).
18. Stollberg, M.: Reasoning Tasks and Mediation on Choreography and Orchestration in WSMO. In Proc. of the 2nd International WSMO Implementation Workshop (WIW '05). Innsbruck, Austria. (2005).
19. Zaremba, M., Moran, M., Haselwanter, T., Lee, H.K. and Han, S.K.: D13.4v0.3. WSMX Architecture WSMX. Available at http://www.wsmo.org/TR/d13/d13.4/v0.3/. (2005).
20. Russell, N., ter Hofstede, A.H.M., van der Aalst, W.M.P. and Mulyar, N.: Workflow Control-Flow Patterns: A Revised View. BPM Center Report BPM-06-22, BPMcenter.org (2006).
21. Krummenacher, R., Hepp, M., Polleres, A., Bussler, C. and Fensel, D.: WWW or What is Wrong with Web Services. In Proc. of 3rd European Conf. on Web Services (ECOWS '05) (2005).
22. Berbers-Lee, T., Hendler, J. and Lassila, O.: The semantic web. Scientific America 284: (2001) 34–43.
23. Haselwanter, T., Kotinurmi, P., Moran, M., Vitvar, T. and Zaremba, M.: WSMX: A Semantic Service Oriented Middleware for B2B Integration. In proc. of the 4th International Conference on Service Oriented Computing (ICSOC '06), Chicago, USA (2006).