Modeling and Analysing Web Services Protocols

Julien Ponge

ponge@isima.fr – http://www.isima.fr/ponge/
Laboratoire LIMOS, ISIMA – Campus des Cézeaux
63173 Aubière cedex, France

Abstract. Web services technology is emerging as the main pillar of service-oriented architectures (SOA). This technology facilitates application integration by enabling programmatic access to applications through standard, XML-based languages and protocols. While much progress has been made toward providing basic interoperability among applications, there are still many needs and unexploited opportunities in this area. In particular, services in SOAs require richer description models than object or component interfaces. This is due to the loose coupling inherent in SOAs and therefore to the fact that services are developed independently of clients. Hence, service descriptions need to include all the information needed by clients to understand if they can interact with a service and how. We outline in this paper a novel approach developed as part of a PhD research work, based on web services protocols descriptions that allows for a high-level analysis based on operators.

1 Introduction

There is still a lot to be done to simplify service development and interaction. In particular, an important aspect of Web services that affects interoperability is that services are loosely-coupled, that is, are not developed only to interact with specific clients but are meant to serve the needs of many different clients, possibly developed by different teams or even different companies. Hence, developers of client applications need to be aware of all functional and non-functional aspects of a service to be able to understand if they can/need inter-operate with a service and how to develop clients that can interact correctly with the service. For this reason, service descriptions should be richer than "just" descriptions of interfaces as in conventional middleware. Specifically, it is commonly accepted that a service description should include not only the interface, but also the *business protocol* supported by the service, i.e., the specification of possible message exchange sequences (conversations) that are supported by the service as well as other useful abstractions (temporal constraints, transactions, vendor policies and so on).

The PhD research work outlined in this paper aims at developing a novel solution to simplify the web services life-cycle management by providing a framework for modeling and analysing web services described as extended protocols. This work is part of a larger effort materialized by the ServiceMozaic platform [1] that enables to support design, development and management of web services in a model-driven CASE tools set environment. The paper first outlines this general framework, then reviews the state of

the art in the field of web services modeling and management. We then expose what we have done so far by considering the temporal constraints abstraction to perform services compatibility and replace-ability analysis. Finally, we conclude and provide directions for future work.

2 Toward a model-driven framework for web services

Protocols specifications describe the external behaviors of services, making them essential for developers willing to create clients that can interact correctly with services. We argue that protocols specifications can considerably simplify web services life-cycle management. For example, during web services development, protocols of clients and providers (possibly obtained from normative efforts such as RosettaNet[1]) can be analyzed to identify which conversations can be carried out between two services, therefore reducing potential runtime errors and suggesting possible modifications to improve the compatibility with the service. Protocol analysis and management can also provide valuable help to support *change support and evolution*. Indeed, it could enable the eased identification of the modifications required by clients at the protocol level when services protocols change. There are also promising applications like automated exceptions handling, compliance verification, and static or dynamic binding, all by taking advantage of the protocols descriptions. To realize such benefits, we aim at developing a conceptual framework, and the associated CASE tool environment, named ServiceMozaic, that enables to support design, development and management of web services. Our primary goal is to develop a model-driven framework within which whole, or at least, central parts of web services are generated and managed from models. The related research issues are the following.

- *Protocol modeling.* We built our framework upon an extended protocol model with formal semantics that allows richer services descriptions. Beside its ability to describe messages choreography constraints (i.e., the legal messages exchange sequences), the proposed model includes relevant abstractions such as temporal constraints, that enables users to better understand the external behavior of services. To avoid creating a model that would be either too simplistic, or too complex, we developed it upon an analysis of real-world e-commerce portals in order to identify the abstractions that can be indeed useful for practitioners.
- *Protocol analysis and management.* We target three types of protocols analysis, namely *compatibility*, *replace-ability* and *consistency* analysis. Compatibility analysis consists in checking whether two services can interact correctly, based on their protocol definitions (i.e., whether a conversation can take place between the considered services). In turn, replace-ability analysis refers to the verification of whether two protocols can support the same set of conversations (e.g., a service can replace another one in general, or when interacting with specific clients). Finally, consistency analysis is related to supporting changes in protocols and their impact on requesters but has not been further investigated.

[1] See `http://www.rosettanet.org/`

- *Protocol algebra and protocol management operators.* A distinctive feature of our approach lies in the definition of operators to query, analyze, and transform protocols. Such operators are the key to carry out most of the features described above.
- *Protocols adapters and code generation.* Theses two issues are part of the Service-Mozaic platform but won't be further detailed in this paper. Briefly, adapters [2] can be created when compatibility (resp. replace-ability) analysis between two protocols reveals that there exist some mismatches that make them not fully compatible (resp. replaceable). Adapters allow to enhance the compatibility / replace-ability level in such situations.

3 State of the art

Tools supporting web services development today are mainly concerned with interoperability issues at the lower levels of the web services stack, like the mappings from WSDL descriptions to Java/C# source code and vice-versa, making two SOAP-based systems talk to each other. Similarly, the existing standards in the higher-level services descriptions such as BPEL4WS, WSCI or WSCL proved to be more concerned with implementation aspects than enabling the kind of formal analysis that we envision. Indeed, the importance of formal analysis of web services protocols in terms of automated support for services interoperability at the business protocol level has been discussed in recent papers: [1, 3–5]. Several efforts recognize aspects of protocol specification in component-based models [6, 7]. They provide models (e.g., pi-calculus based languages for component interface specifications) and algorithms (e.g., compatibility checking) that can be generalized for use in web services protocol specifications and management. Indeed, various efforts in the general area of formalizing web services description and composition languages emerged recently [5, 8]. However, in terms of managing the web services development life-cycle, technology is still in the early stages. Consistency analysis should have some interesting links with existing software engineering techniques such as refactoring [9].

Our approach, based on a protocol algebra and protocol operators is novel in the field. Specifically, it should be more fine-grained than the approaches mentioned above when doing compatibility and replace-ability analysis. We believe that to some extent, the development of a protocol algebra for formal services descriptions analysis implemented inside a larger CASE tool, based on the identification of abstractions needed by practitioners [10], can have the same impact that relational algebra had for relational databases.

4 Web services protocols modeling and analysis

4.1 A model extended with temporal constraints

Our model is based on the *web services business protocol* proposed in [10, 1], which is built upon the traditional state machine formalism to represent messages choreography constraints. States represent the different phases that a service may go through during its interaction with a requester to the provider and vice-versa. A message corresponds

to the invocation of a service operation or to its reply. Hence, each state identifies a set of outgoing transitions, and therefore a set of possible messages that can be sent or received. Each transition is labeled with a message name followed by the message polarity, that is, whether the message is incoming or outgoing. The protocols are deterministic, that is to say, they have one initial state and, each state cannot provide more than one outgoing transition labeled with the same message. The model also supports the notion of *final states*, which correspond to the end of a successful conversation, in the sense that the messages exchanges between the provider and the requester is over on both sides. Briefly, the reason for using state machines as the basis for the model is because it is familiar to users, it is suitable to described reactive behaviors, and the notion of states is useful to perform services execution monitoring.

We have extended the model (called *timed web service business protocol*) to cater for temporal abstractions in [11, 12]. Transitions can become *timed transitions* when carrying temporal constraints. We identified two kinds of timed transitions. The first one (which we called *C-Invoke constraints*) relates to a time window during which the related operation can be triggered explicitly by either the provider or the requester. The other kind of transition that we identified corresponds to *timed implicit transitions* that can automatically occur once a certain date and time has been reached. We called this type of constraints *M-Invoke*. A proper discussion on the formal semantics of the model would require too much space to fit in this paper, but we can recall that they are based on the notion of *timed execution traces*, inspired by timed automata [13].

4.2 Temporal compatibility and replace-ability analysis

Reactive systems have been widely studied, notably to the benefit of software and hardware verification. For analysis purposes, two formal frameworks are of interest. The first one is the temporal logics theory [14]. It allows for expressing complex requirements expressed as formulas that have to be satisfied on reactive systems modeled as automata. For example, a complex safety property (*"the event will never happen"*) or a liveness property (*"the event will happen"*) can be described within this framework. It appears that we won't need temporal logics and their timed extensions (such as [15]) as we need to define simple timed constraints. The other framework is the timed automaton [13] which provides a lot of interesting automata classes, notably the event-recording automaton that has some valuable decidability and complexity properties on verification techniques. We have developed mappings from/to timed web services business protocols that are useful from a formal point of view. However, we try not to use straight the existing timed automata verification techniques which have been developed with the general cases in mind and are thus more demanding than the ad-hoc algorithms that we can develop for timed protocols. What's more, traditional verification techniques lack the identification of partial compatibility and replace-ability, which is an important contribution of this work. We have defined *classes* to identify different levels of compatibility and replace-ability, as well as *operators* that can be applied to protocols definitions to assess the level of compatibility and replace-ability. These classes can be characterized by using and combining 3 operators (the *timed compatible composition*, the *timed intersection* and the *timed difference*), for which we have a more detailed def-

inition as well as polynomial-time algorithms in [11]. We introduce the compatibility and replace-ability classes below.

- *Partial compatibility* (or simply, compatibility): A protocol P_1 is partially compatible with another protocol P_2 if there are some executions of P_1 that can inter-operate with P_2, i.e., if there is at least one possible conversation that can take place among two services supporting these protocols
- *Full compatibility*: a protocol P_1 is fully compatible with another protocol P_2 if all the executions of P_1 can inter-operate with P_2, i.e., any conversation that can be generated by P_1 is understood by P_2.
- *Protocol equivalence w.r.t. replace-ability*: two business protocols P_1 and P_2 are equivalently replaceable if they can be interchangeably used in any context and the change is transparent to clients.
- *Protocol subsumption w.r.t. replace-ability*: a protocol P_2 is subsumed by another protocol P_1 w.r.t. replace-ability if P_1 supports at least all the conversations that P_2 supports. In this case, protocol P_1 can be transparently used instead of P_2 but the opposite is not necessarily true.
- *Protocol replace-ability with respect to a client protocol*: A protocol P_1 can replace another protocol P_2 with respect to a client protocol Pc if P_1 behaves as P_2 when interacting with a specific client protocol Pc.
- *Protocol replace-ability with respect to an interaction role*: Let P_R be a business protocol. A protocol P_1 can replace another protocol P_2 with respect to a role P_R if P_1 behaves as P_2 when P_2 behaves as P_R. This replace-ability class allows to identify executions of a protocol P_2 that can be replaced by protocol P_1 even when P_1 and P_2 are not comparable with respect to any of the previous replace-ability classes.
- *Partial protocol replace-ability*: Partial replace-ability is when there is replace-ability but only for some conversations and not others. For example, we have partial replace-ability with respect to a client protocol when protocol P_1 can replace another protocol P_2 in at least some of the conversations that can occur with another P_C protocol.

5 Conclusion

After one year of work, we have proposed an extension of the web services business protocols model proposed in [1] to cater with temporal abstractions in [12, 11]. We proposed a novel approach by characterizing the temporal compatibility and replace-ability classes by using protocols operators for which we have efficient algorithms. We also have a link to the timed automata [13] theory that allows us to assess from a formal point of view that it has interesting properties. The implementation of the web services business protocol model has been done and will be soon extended with temporal abstractions. We have also implemented an editor for protocols in the ServiceMozaic platform as a GEF-based Eclipse plug-in[2].

Future work includes focusing on the third kind of protocols-based analysis: consistency analysis. We will also need to take into account multi-protocols choreography

[2] See `http://www.eclipse.org/` and `http://www.eclipse.org/gef/`.

analysis (the current work is focused on the analysis between a service provider and a requester). Finally, other useful abstractions such as transactional properties will provide another promising area of investigation.

References

1. Farouk Toumani, Boualem Benatallah, Fabio Casati: Analysis and Management of Web Services Protocols. DKE, Special issue from ER'04 (2004)
2. Benatallah, B., Casati, F., Grigori, D., Nezhad, H.R.M., Toumani, F.: Developing adapters for web services integration. In Springer-Verlag, ed.: Procs of CAiSE'05. Lecture Notes in Computer Science (2005)
3. Bordeaux, L., Salaun, G., Berardi, D., Mecella, M.: When are two Web Services Compatible? In: VLDB TES'04, Toronto, Canada. (2004)
4. Hull, R., Benedikt, M., Christophides, V., Su, J.: E-services: a look behind the curtain. In: Proc. 22th Principles of Database Systems (PODS'03), San Diego, CA, USA, ACM (2003) 1–14
5. Bultan, T., Fu, X., Hull, R., Su, J.: Conversation specification: a new approach to design and analysis of e-service composition. In: WWW 2003, Budapest, Hungary, ACM (2003) 403–410
6. Canal, C., Fuentes, L., Pimentel, E., Troya, J., Vallecillo, A.: Adding Roles to CORBA Objects. IEEE Trans. Software Eng **29** (2003) 242–260
7. Yellin, D., Storm, R.: Protocol Specifications and Component Adaptors. ACM Trans. Program. Lang. Syst. **19** (1997) 292–333
8. Mecella, M., Pernici, B., Craca, P.: Comatibility of e-services in a cooperative multi-platform environment. In: VLDB-TES'01, Springer (2001)
9. Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D.: Refactoring – Improving the Design of Existing Code. Addison Wesley (1999)
10. Benatallah, B., Casati, F., Toumani, F., Hamadi, R.: Conceptual Modeling of Web Service Conversations. In: Procs of CAiSE'03. Volume 2681 of LNCS., Klagenfurt, Austria, Springer (2003) 449–467
11. Boualem Benatallah, Fabio Casati, Julien Ponge, Farouk Toumani: Compatibility and replaceability analysis for timed web service protocols. In: Proceedings of BDA 2005, Saint-Malo, France. (2005)
12. Boualem Benatallah, Fabio Casati, Julien Ponge, Farouk Toumani: On Temporal Abstractions of Web Services Protocols. In: Proceedings of CAiSE Forum 2005, Porto, Portugal. (2005)
13. Rajeev Alur, David L. Dill: A theory of timed automata. Theoretical Computer Science (1994) 183–235
14. Pnueli, A.: The temporal logic of programs. In: FOCS. (1977)
15. Alur, R., Courcoubetis, C., Dill., D.: Model-checking in dense real-time. Information and Computation **1** (1993) 2 – 34