ReMan: a Pro-active Reputation Management Infrastructure for Composite Web Services

Domenico Bianculli Walter Binder
University of Lugano
Faculty of Informatics
Lugano, Switzerland
domenico.bianculli@lu.unisi.ch
walter.binder@unisi.ch

Mauro Luigi Drago Carlo Ghezzi
Politecnico di Milano
DEEP-SE group - DEI
Milano, Italy
drago@elet.polimi.it
carlo.ghezzi@polimi.it

Abstract

ReMan *is a reputation management infrastructure for composite Web services. It supports the aggregation of client feedback on the perceived QoS of external services, using reputation mechanisms to build service rankings. Changes in rankings are pro-actively notified to composite service clients to enable self-tuning properties in their execution.*

1. Introduction

Web services are the de-facto standard for the development of distributed applications executing in open environments. Web services are usually composed by means of languages like BPEL [12] to provide added-value services. Composite services, typically built from third-party services, have to adapt to the open, dynamically changing environment where remote services may fail or new services may be offered at any moment. Thus, one key feature of composite services is the ability to adapt their service bindings so as to leverage the best performing services currently available in the evolving service market.

This step can be accomplished by using a reliable and efficient mechanism to get service rankings from the execution environment. To this end, *reputation mechanisms* have been proposed [15]. They collect client ratings on experienced service behavior to compute the actual Quality-of-Service (QoS) delivered to clients and to rank functionally equivalent services accordingly. However, none of the current environments for the execution of composite services, such as BPEL engines, transparently integrates a reputation mechanism. They often require to manually modify a composite service for interacting with a reputation mechanism so as to report feedback on service interactions and to dynamically choose the most efficient services. This requirement is cumbersome and error-prone. Furthermore, an appropriate monitoring infrastructure is needed to assess experienced service functionality and QoS.

In prior work [5], some of the authors proposed an architecture for automated, dynamic, pro-active, and transparent maintenance and improvement of composite services, promoting an incentive-compatible reputation manager to share reliable service quality information among clients. However, the requirements posed by the theoretical model made the implementation impractical. In [4], we proposed a new architecture for reputation management, enabling the transparent integration of reputation mechanisms in standard execution environments for composite services. The new architecture is characterized by the following features:

1. It provides a mechanism for assessing service behavior and ranking functionally equivalent services (i.e., with compatible WSDL interfaces) based on past interactions with these services by other clients.
2. It supports user notifications when particular reputation-related events occur, allowing for an early discovery of possible failure situations.
3. It ensures reputation-aware execution of composite services in a way that is completely transparent to the programmer, who can concentrate exclusively on the functional aspects of the composite service.
4. It allows for an open and extensible platform supporting a high degree of customization of the way service reputation is computed.

This paper presents ReMan, our implementation of the architecture described in [4], built from state-of-the-art middleware technologies. ReMan is available at `http://www.inf.unisi.ch/phd/bianculli/research/reman/`.

2. REMAN architecture

In this section, we describe the software architecture of REMAN, both at the server- and at the client-side, which is also depicted in Figure 1.

2.1. Server-side REMAN

The server-side architecture of REMAN comprises three main components: the *enhanced registry*, the *reputation manager*, and the *subscription manager*.

Enhanced Registry. It is a UDDI-compliant registry with standard interface for service publishing and service discovery, enriched with support for queries on QoS estimations of registered services. The registry can be queried either by providing a specific service TModel or by providing the concrete service location and the WSDL interface it complies to.

Reputation Manager. It provides functionalities to manage the services registered for reputation assessment and to estimate their QoS. It receives feedback reports from service clients and UDDI-related events (e.g., registration of a new service) from the *Enhanced Registry*, and consequently updates the underlying data storage.
The computation of service reputation is performed by this component, according to the installed *reputation policy* — which represents our abstraction for an algorithm that estimates service reputation — at specific time instants, as planned by a scheduler that tries to avoid minor fluctuations in the reputation estimates, which could trigger unnecessary notifications.

Subscription Manager. It provides functionalities to notify service consumers when reputation-related events occur and to manage the subscriptions to these events.
REMAN supports two event types: *reputation decrease* and *availability of a service with better reputation*.
The former event is fired when the *Reputation Manager* communicates that the reputation of a service has dropped below a certain threshold. Service users are thus notified of a possible failure condition by means of these messages, so that countermeasures can be taken. Upon subscription, each service client specifies the services for which it should receive notifications on reputation decrease and, for each of them, the reputation threshold.
The latter event is used to notify service clients when the set of the "best" services (i.e., the ones with the currently best reputation) compliant with a particular specification changes. By means of these notifications, we let service clients always know which are the best services available on the service market such that when a possible failure occurs they may rebind to another service, which exhibits better behavior. Service clients subscribe to these events by specifying the WSDL interface they are interested in.

2.2. Client-side REMAN

At the client-side, the architecture comprises three components:

Monitor. It monitors the behavior of external services used by the BPEL service client, by checking the functional and non-functional properties attached to the BPEL process. The properties can be expressed in WS-CoL [2] and in ALBERT [1]. The following sample WS-CoL property specifies that a service invocation should return a positive value (saved in variable Loan/amount), with a response time less than 5s: `$RespTime < 5s and $Loan/amount > 0`.

Reputation Feeder. It provides methods to collect feedback reports and to send them to the *Reputation Manager* on the server.

Event Manager. It provides functionalities to subscribe to reputation-related events and to react to such notifications.

3. REMAN at work

This section describes a typical operation scenario of REMAN; the referenced exchanged messages are depicted in Figure 1.

1. Service providers publish their services (e.g., services A and B) using the UDDI-compliant interface offered by the *Enhanced Registry* (message P1). Internally, the *Enhanced Registry* notifies the *Reputation Manager* that a new service has been registered, and thus that a default reputation should be assigned to it (message P2). The *Enhanced Registry* also notifies the *Subscription Manager* so that it can notify interested service clients of the availability of a new service (message P3).

2. When service clients deploy their business processes into the BPEL engine, the client module of the reputation infrastructure logs into the server module (message D1), in order to get access credentials for subsequent communications. Service clients communicate the selected service bindings to the server using the *Event Manager* (message D2); in this way, clients subscribe to events related to (the type of) services they use. For example, the BPEL service depicted in Figure 1 will communicate to the *Reputation Manager* its bindings to services A and B, used within the business process by the activities A1 and A3.

3. During execution, each time a client uses an external service, the built-in monitor checks for the property associated with the interaction. The result of the evaluation is sent to the *Reputation Feeder* (message F1), which generates a boolean feedback corresponding to the evaluation of the logical formulae that constitute the monitoring property. This message is then sent (message F2) to the *Reputation Manager*, on the server module of REMAN.

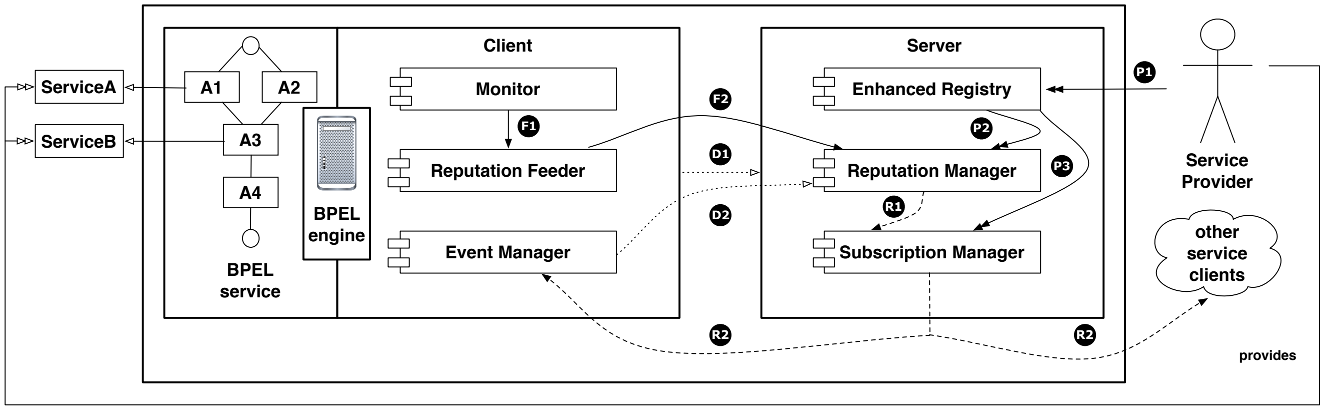

Figure 1. REMAN architecture and interactions between its components

4. After collecting reputation feedback reports, the *Reputation Manager* updates the reputation estimation of the services registered in the system. The reputation is generated by computing the ratio of the number of positive feedbacks and the total number of feedbacks received until the computation of the estimation is triggered by the system. Whenever the *Reputation Manager* computes a new value of the reputation of a service, it notifies the *Subscription Manager* (message R1). The latter can then either communicate (message R2) to all subscribed clients that the reputation of a service dropped below a certain threshold, or it can notify them that a new service implementing a certain WSDL interface and with a better reputation became available.

4. Implementation

REMAN has been entirely implemented as a JavaEE compliant application. The *Reputation Manager* and the *Subscription Manager* have been implemented by means of both stateless session beans and message-driven ones. Most of the functionalities of the *Enhanced Registry* have been implemented by means of stateless session beans; standard UDDI services are instead provided by means of Web service beans and by the Grimoires UDDI registry [1], which has also been extended to support notifications about changes in the database of UDDI entities.

Some operations, such as finding the best services compatible with a certain interface, require a notion of service equivalence. We implemented a component that performs the analysis of WSDL documents — associated with UDDI TModels — and generates sets of compatible services.

REMAN has been designed in an open and extensible way, so as to support different methods for computing service reputation, in the form of *reputation policies* provided as plugins for the *Reputation Manager*. The default reputation policy plugin in our reference implementation estimates the reputation of each single service published in the infrastructure by using a *bayesian reputation approach with binary ratings* [6].

At the client-side, service monitoring is performed with Dynamo [3], built on top of the ActiveBPEL engine[2], and extended to support sending feedback when a monitoring property is evaluated.

In terms of security, every message exchanged in the system is secured against tampering and replay attacks; access to the system is granted by means of a public key mutual authentication algorithm. The inclusion of mechanisms that encourage clients to honestly report feedback [7] is still under development.

5. Related work

The agent-based trust framework for service selection described in [11] relies on the conceptual model for Web service reputation proposed in [10]. In contrast to REMAN, the authors use a different architectural style, where a software agent is attached to each Web service; the agents are in charge of querying and reporting service reputation. Each service client builds its reputation of services based on the *local* information provided by its neighbors.

A QoS-based service selection model is presented in [8]. Like REMAN, the model takes into account the feedback from users; moreover it also supports user-defined QoS selection criteria. However, with respect to REMAN, it is neither pro-active (because variations of service reputation are not disseminated to other service clients, which could benefit from this information), nor transparent (since service requesters are required to support specific mechanisms for ad-hoc execution monitoring and feedback reporting).

[1]`http://www.grimoires.org`.

[2]`http://www.activevos.com/`.

A service recommendation system is proposed in [9]. In this system, clients rate services by using a comparative matrix containing the QoS values advertised by the provider, and the QoS values measured at run time. However, the system does not use Web service standards for service discovery and selection, but relies on ontology-based descriptions. Moreover, user feedback reporting is not automated.

In [14], the authors describe a method to collect monitoring data from clients and to use this information for service recommendations. However, the supported QoS metrics are limited to client side performance, such as throughput and response time.

A collaborative filtering approach to derive prediction of QoS of Web services that were not used yet, based on the experience of consumers of similar services, is proposed in [13]. However, the whole approach is poorly integrated in the execution environment and it is neither fully automated nor transparent. Moreover, it supports only the prediction based on the evaluation of timeliness-related QoS properties.

The approach described in [16] adopts a point of view that is complementary to ours. The reputation of a composite service is derived based on the reputation of the single services used within the composition. The reputation mechanism used to compute the reputation of the single services is similar to ours.

Finally, [17] proposes a framework for run-time service discovery in both pull and push modes, based on structural and behavioral models of services, as well as quality and contextual constraints. When the framework operates in push mode, it shares similar functionalities with REMAN; however services selection is not driven by any reputation mechanisms.

6. Conclusion

This paper presents REMAN, a reputation management infrastructure that supports pro-active service selection for composite Web services, by allowing them to bind to the best available services available in the evolving service market. This feature, integrated in existing state-of-the art run-time infrastructures and compatible with industry standards, fosters dynamic adaptability and self-tuning properties in the execution of composite services.

7. Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement no. 215483-S-CUBE and ERC (IDEAS programme) grant agreement no. 227977-SMScom.

References

[1] L. Baresi, D. Bianculli, C. Ghezzi, S. Guinea, and P. Spoletini. Validation of web service compositions. *IET Softw.*, 1(6):219–232, 2007.

[2] L. Baresi and S. Guinea. Towards dynamic monitoring of WS-BPEL processes. In *Proceedings of ICSOC'05*, volume 3826 of *LNCS*, pages 269–282. Springer, 2005.

[3] L. Baresi, S. Guinea, and L. Pasquale. Self-healing BPEL processes with Dynamo and the JBoss rule engine. In *Proceedings of ESSPE'07*, pages 11–20. ACM, 2007.

[4] D. Bianculli, W. Binder, L. Drago, and C. Ghezzi. Transparent reputation management for composite web services. In *Proceedings of ICWS 2008*, pages 621–628. IEEE Computer Society, 2008.

[5] D. Bianculli, R. Jurca, W. Binder, C. Ghezzi, and B. Faltings. Automated dynamic maintenance of composite services based on service reputation. In *Proceedings of ICSOC'07*, volume 4749 of *LNCS*, pages 449–455. Springer, 2007.

[6] A. Josang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decis. Support Syst.*, 43(2):618–644, 2007.

[7] R. Jurca and B. Faltings. Minimum payments that reward honest reputation feedback. In *Proceedings of EC'06*, pages 190–199. ACM, 2006.

[8] Y. Liu, A. H. Ngu, and L. Z. Zeng. QoS computation and policing in dynamic web service selection. In *Proceedings of WWW Alt. '04*, pages 66–73. ACM, 2004.

[9] U. S. Manikrao and T. V. Prabhakar. Dynamic selection of web services with recommendation system. In *Proceedings of NWESP'05*, pages 117–121. IEEE Computer Society, 2005.

[10] E. M. Maximilien and M. P. Singh. Conceptual model of web service reputation. *SIGMOD Rec.*, 31(4):36–41, 2002.

[11] E. M. Maximilien and M. P. Singh. Toward autonomic web services trust and selection. In *Proceedings of ICSOC'04*, pages 212–221. ACM, 2004.

[12] OASIS. Web Service Business Process Execution Language Version 2.0 Specification, 2007.

[13] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei. Personalized qos prediction for web services via collaborative filtering. In *Proceedings of ICWS 2007*, pages 439–446. IEEE Computer Society, 2007.

[14] N. Thio and S. Karunasekera. Web service recommendation based on client-side performance estimation. In *Proceedings of ASWEC'07*, pages 81–89. IEEE Computer Society, 2007.

[15] Y. Wang and J. Vassileva. A review on trust and reputation for web service selection. In *Proceedings of ICDCS'07 Workshops*, pages 25–32. IEEE Computer Society, 2007.

[16] S. J. H. Yang, J. S. F. Hsieh, B. C. W. Lan, and J.-Y. Chung. Composition and evaluation of trustworthy web services. In *Proceedings of BSN'05*, pages 5–12. IEEE Press, 2005.

[17] A. Zisman, J. Dooley, and G. Spanoudakis. Proactive runtime service discovery. In *Proceedings of SCC 2008*, pages 237–245. IEEE Computer Society, 2008.