Adapt: Towards Autonomic Web Services*

Alberto Bartoli[1], Ricardo Jiménez-Peris[2], Bettina Kemme[3], Cesare Pautasso[4], Simon Patarin[5], Stuart Wheater[6], and Simon Woodman[7]

[1] Univ. di Trieste, Italy
[2] Univ. Politecnica de Madrid, Spain
[3] McGill Univ., Canada
[4] ETH Zurich, Switzerland
[5] Univ. di Bologna, Italy
[6] Arjuna Technologies, United Kingdom
[7] Univ. of Newcastle, United Kingdom

1 Introduction

Organizations are increasingly using the Web not only to sell products and deliver information, but also for providing their services to businesses and individual customers. Typically, the provision of such services by organizations requires the construction of applications that integrate existing enterprise information systems to offer new business functions. Organizations need to ensure that these services are available, scalable and also autonomic to guarantee that user interactions are promptly processed even under highly volatile conditions. In most cases, organizations use application servers with a multi-tier architecture for the delivery of their services. In the following, we will refer to services not relying on other web services as Basic Services (BSs). The presence of a wide variety of BSs over the Internet has created an exciting new business opportunity of providing value added, inter-organizational services by composing multiple BSs into new Composite Services (CSs). The Adapt project aims to provide autonomic support for both basic and composite web services. In the following sections we describe autonomic support for both kinds of web services.

2 Self-Descriptive Web Services

In general *functional properties* of a web service are described by means of a WSDL (Web Service Description Language) document. Within Adapt, it is also possible to describe the *non-functional properties* of the service by means of a *service specification language* developed for this purpose. This description enables a rich composition of services in which properties of a CS are determined out of the properties of its constituent services and validated against them. We have identified three families of non-functional attributes that may be of interest for compositions: 1) Attributes related to transactional capabilities of web services; 2) Attributes regarding sequencing constraints on web service invocations (conversations) [13]; 3) Attributes dealing with web service performance.

* The Adapt project is an FP5 IST project funded by European Commission under contract IST-2001-37126. http://adapt.ls.fi.upm.es/adapt.htm

The first and the second set of attributes are static in the sense that their specification does not change after deployment. It is useful to publish static attributes at deployment time. This allows a CS to use the static attributes at composition time to derive, and verify, the properties of the composition. These attributes are described in the same WSDL document describing the functional properties of the service, using the extensibility features of WSDL. The third set of attributes is dynamic: performance values, like response time, throughput, and availability that change continuously during service execution. These dynamic attributes are useful at run-time to choose the most appropriate service with respect to its actual performance. Unlike the above static attributes, these are associated with dedicated web service operations: the CS can invoke these web services to find out the current performance values of the web service.

The second set of attributes is used by the analysis tool to verify automatically that the structure of the composition is free of deadlocks and livelocks such as infinite loops, unreachable execution branches etc. It also verifies that the messages sent and received from the process to each component service form a valid conversation, with respect to that services sequencing constraints [13].

3 Autonomic Support for Basic Services

The support for basic web services adopts a three-tier J2EE approach. The three tiers considered are: web tier (web service components), application server tier (EJBs), and database. Our middleware for J2EE-based BSs uses *replication* as the key technique for providing adaptability. It allows replicating the application server and/or the database server.

The support for J2EE replication is organized into two parts: i) a generic replication framework for J2EE-based web services in charge of intercepting client requests, EJB invocations, and replies [1]; ii) specific algorithms for autonomic replication [14, 3]. With this approach, that we use at the middle tier, the task of developing a replication algorithm is decomposed into two portions: the framework itself, which handles all the detailed interactions with the underlying application server code, and the specific replication algorithm, which is written in terms of a high-level API. The framework only requires implementing once per platform, allowing the developers to concentrate on the relevant details of the specific**?**replication algorithm. The framework has been implemented for the JBoss open-source application server and the Axis SOAP engine. The framework is based on JBora (which is built on top of Spread) that also introduces adaptive message batching [2].

Concerning the database layer, two approaches have been developed based on PostgreSQL: Middle-R [9, 7, 4] a middleware approach to database replication and Postgres-R [15] that extends the database code with the replication logic. Autonomic support at the database layer focuses on load control [8], dynamic load balancing [8], and online recovery [6, 5]. At the local level, load control enables to regulate automatically the multi-programming level (MPL, the number of allowed concurrent transactions) to attain the maximum throughput. At the global level, dynamic load balancing enables to distribute the load across replicas. In order to maintain the level of availability, failed replicas should be recovered or substituted by new ones. Offline recovery would result in a loss of availability during recovery. For this reason, in Adapt, replicas are recovered online without stopping request processing.

4 Autonomic Support for Composite Services

Workflows provide a good abstraction to model the composition of web services. A workflow process is composed of a set of service invocations (or tasks) and its structure defines the control and data flow dependencies between them. With this approach, it becomes possible to give a high-level description over the partial order of invocation of the services and their interactions while maintaining well-defined, executable semantics. In Adapt, a process can be modelled with different syntaxes: visual [10] and XML-based.

In Adapt, the service composition language has been designed to allow the specification of the structure of a composite service at a level of abstraction which allows the composite service designer to concentrate on modelling the correct functional behaviour of the process.

Due to the properties of the distributed setting in which composite services are deployed, it is essential that a composite service can be modified dynamically to reflect changes in the underlying environment. Adaptability is thus a very important property of a composite service that can be expressed along several orthogonal dimensions:

- Dynamic binding. The references linking the composite service to its component services are evaluated at the latest possible time. Thus, it becomes possible to retarget some of the services to be invoked dynamically, ensuring the forward progress of the composition [12].
- Fault-tolerance. The composite service should appropriately handle application-level failures that are encountered during the invocation of its basic services. Similarly, if a service becomes unavailable it should be possible to retry the invocation using a backup service provider.
- Resolution of interface mismatches. Bottom-up composition may result in incompatibilities in the data models of the basic services to be integrated. Thus, a suitable mechanism to adapt mismatching services to fit with one another is required.
- Advanced Transactional Semantics. With this, compositions can adapt to exceptional conditions and remain consistent.

Introducing adaptability in a composite service through the composition language is not enough, as adaptability needs to be appropriately supported by the underlying composition infrastructure. In this regard, the Adapt distributed engine for executing compositions can be adapted to provide different levels of performance in terms of both scalability and reliability. In [11] we show that replication can be applied to key components of the engine in order to increase the overall system's throughput as the engine runs over a cluster of computers. Similarly, we evaluated the cost of providing reliable execution of the composition by comparing different strategies for the adding persistence to the engine.

5 Acknowledgments

Many people contributed to the Adapt project: G. Alonso, E. Antoniutti, O. Babaoglu, B. Biörnstad, D. Ingham, D. Jönsson, Y. Lin, M. Little, V. Maverick, J. Milan, D. Palmer, M. Patiño-Martínez, F. Perez, M. Prica, M. Saheb, J. Salas, D. Serrano, S. Shrivastava, J. Vuckovic, H. Wu.

References

1. Ö. Babaoglu, A. Bartoli, V. Maverick, S. Patarin, J. Vuckovic, and H. Wu. A Framework for Prototyping J2EE Replication Algorithms. In *DOA*, 2004.
2. A. Bartoli, C. Calabrese, M. Prica, E. Antoniutti, and A. Montresor. Adaptive message packing for group communication systems. In *Works. on Reliable and Secure Middleware*, 2003.
3. A. Bartoli, M. Prica, and E. Antoniutti. A replication framework for program-to-program interaction across unreliable networks and its implementation in a servlet container (in press). *Concurrency and Computation: Practice and Exper.*, 2005.
4. R. Jiménez-Peris, M. Patiño-Martínez, and G. Alonso. Non-Intrusive, Parallel Recovery of Replicated Data. In *IEEE SRDS*, 2002.
5. R. Jiménez-Peris, M. Patiño-Martínez, G. Alonso, and B. Kemme. Are Quorums an Alternative for Data Replication. *ACM Trans. on Databases*, 28(3), 2003.
6. B. Kemme, A. Bartoli, and O. Babaoglu. Online Reconfiguration in Replicated Databases Based on Group Communication. In *DSN*, 2001.
7. Y. Lin, B. Kemme, M. Patiño-Martínez, and R. Jiménez-Peris. Middleware based data replication providing snapshot isolation. In *ACM SIGMOD Conf.*, 2005.
8. J. M. Milán, R. Jiménez-Peris, M. Patiño-Martínez, and B. Kemme. Adaptive middleware for data replication. In *ACM/IFIP/USENIX Middleware*, 2004.
9. M. Patiño-Martínez, R. Jiménez-Peris, B. Kemme, and G. Alonso. Consistent Database Replication at the Middleware Level (In Press). *ACM Transactions on Computer Systems*, 2005.
10. C. Pautasso and G. Alonso. Visual composition of web services. In *HCC*, pages 92–99, 2003.
11. C. Pautasso and G. Alonso. JOpera: a Toolkit for Efficient Visual Composition of Web Services. *J. of Electronic Commerce*, 9(2), 2004.
12. C. Pautasso and G. Alonso. The JOpera visual composition language. *J. Vis. Lang. Comput.*, 16(1-2):107–141, 2005.
13. S. Woodman, D. Palmer, S. Shrivastava, and S. Wheater. Notations for the specification and verification of composite web services. In *IEEE EDOC*, 2004.
14. H. Wu, B. Kemme, and V. Maverick. Eager Replication for Stateful J2EE Servers. In *DOA*, pages 1376–1394, 2004.
15. S. Wu and B. Kemme. Postgres-r(si): Combining replica control with concurrency control based on snapshot isolation. In *ICDE*, 2005.