Web Services Based Execution of Business Rules

Rainer Schmidt[1]

[1] University of Cooperative Education Lörrach, Hangstraße 46-50,79539 Lörrach, Germany
Rainer.Schmidt@ba-loerrach.de

Abstract. The inherent distributed structure of business rules shows a high affinity to the execution of business processes across the internet, as needed for e-business and e-commerce. However, doing e-business also requires the integration of an evolving set of heterogeneous services into the business processes. To cope with these requirements, a web service based execution model is proposed.

1 Introduction

The execution of business processes across the internet is the core of e-business and e-commerce. These business processes are not static, but dynamically evolving. Many services have to interoperate seamlessly, such as ordering, billing, calculation services etc. In addition, the set of services is not fixed, because business process changes and extensions require the integration of additional services. The services are scattered around different enterprises and implemented using different platforms, programming languages etc. Due to the fact, that there is no centralized organization, the services are evolving independently. In summary, the execution of business processes means not only the distributed execution of business processes, but also the integration of an evolving set of heterogeneous services.
Business processes can be described by a variety of means for instance objects [FeSi94], [FeSi96], [UML] or events [Sche94]. Describing business processes by business rules [HKMS94] is done by splitting up the business process to patterns, which follow an event-condition-action scheme. Therefore, each rule describes which actions have to be done if a defined event happens and a defined condition is met. Thus, business rules provide a kind of modularization for business processes and offer advantages such as flexibility and reusability. Business process changes can be easily implemented by the exchange of business rules and individual business rules can be easily reused across different business processes.
The inherent distributed nature of business rules make them a perfect fit for the distributed execution of business processes across the internet, due to their distributed nature. However, to support business processes across the Internet, the execution of business rules has also to meet the other requirements identified above. Therefore, business rules need an appropriate execution model and environment to cope with the additional requirements.
Web services [W3WS], [GrSi02] address these requirements, which are not provided by business rules yet. Therefore, this paper will present first ideas how to combine the

processing of business processes using business rules with the capability of web services to integrate heterogeneous and evolving sets of services.

2. Web Services

The goal to integrate evolving sets of heterogeneous services has been already addressed by many approaches, which converge into the concept of middleware. There are different middleware technologies such as object-oriented middleware technologies, for instance CORBA [OMG] or component-oriented middleware, for instance DCOM [Chap96]. Both gave the promise to integrate services in a distributed environment across operating systems, implementation languages and programming paradigms.
However, these approaches failed. Although CORBA created a basic interoperability via IIOP, it does not provide complete interoperability across the different implementations of CORBA by different vendors [Giso01]. Furthermore, technologies such as CORBA or DCOM require a large infrastructure, creating huge efforts in cost and time for their introduction. See [GrSi02], [IBM1] for a detailed discussion.
To avoid such problems, web services follow another approach to provide service integration. Web services use existing internet technologies such as HTTP and XML [XML] as technological basis and therefore are able to use the infrastructure available in many enterprises. By this means, many problems due to incompatibilities can be avoided. Furthermore, web services do not try to create another middleware such as CORBA, DCOM, but reuse the functionality already provided by them.
Web services are implemented by a set of technologies (see [GrSi02] for a complete description). The basic technology of web services is SOAP [W3WS], the simple open access protocol. SOAP specifies how services can interoperate over the Internet, by “wrapping” requests and the responses as XML-documents.
The basic principle of SOAP can be illustrated by a request response sequence, as it is used for remote procedure calls in a client-server relationship (see Figure 1). The client can be implemented in any programming language and platform. The same applies for the server. The soap converter translates the programming language and platform dependent request of the client into a so-called soap request. This is a XML-document, following a schema defined in [W3WS]. This document is sent to the receiver of the request, using standard internet protocols such as HTTP or SMTP, facilitating the traversal of firewalls. On the server side, another soap converter decodes the XML-document representing the request into the format required by the server. After the server has processed the request, the converter encodes the result as an XML-document, a so-called SOAP-response that is sent back, converted for the client, and delivered to him.

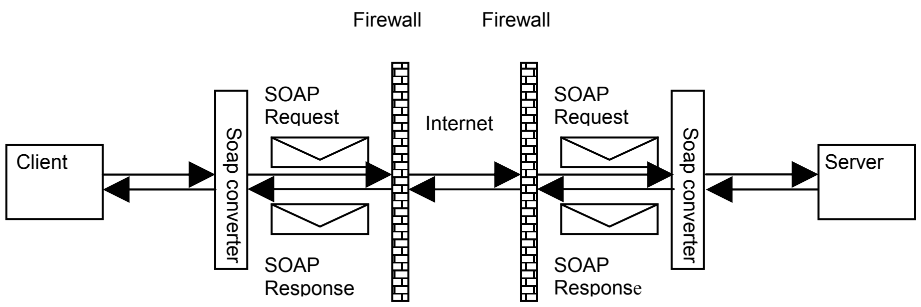

Figure 1: SOAP request-response sequence

A common misunderstanding about SOAP is that the remote procedure call style described above is the only one supported by SOAP. However, SOAP is really specified as a one-way, stateless protocol [W3WS], which can be used to create rather complicated processing protocols. The key to this capability is the structure of the XML-document used for the transport of the SOAP-request, the so-called SOAP envelope (see Figure 2). SOAP envelopes, contain a body for the “pay-load” of the request (e.g. the procedure parameters), but also a header. The header contains meta-information controlling the processing of the information in the body. Typical uses are the definition of the transaction or security context of the information in the body. However, the information in the header can be freely defined. It may contain any information, separated into different entries. The structure and the semantics of each entry may be defined by different XML-namespaces. Thus, a clear separation of the entries is provided.

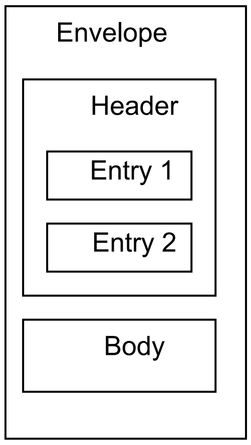

Figure 2: Structure of a SOAP-envelope

This structure of a SOAP envelope can be used to control the processing of information in multiple steps by so-called intermediaries. An intermediary is the addressee of a header entry, which defines the operations to be performed by the intermediary. E.g., entry 1 in Figure 2 may control the processing by intermediary one, entry 2 the processing of intermediary two etc. By use of intermediaries, complex processing protocols may be implemented based on web services.

3. An Execution Model for Business Rules Using Web Services

The capability of SOAP-envelopes to control the processing over multiple intermediaries is the basis for an execution model for business rules using web services. In this execution model, the execution of business processes is done by forwarding an SOAP-envelope between intermediaries, which are responsible for processing one or several business rules. The first processing point is called initiator, the last one addressee. The business rules to be processed are represented by header entries of a SOAP envelope. By defining an execution sequence of the header entries, we can represent the execution sequence of the business rules.

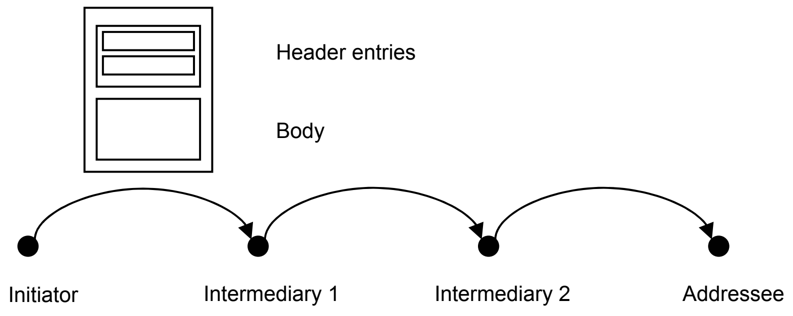

Figure 3: Basic execution model

Because the header entries may use any namespace, business rules represented in RuleML [rule] may be inserted as entries into the header of an SOAP-envelope. Furthermore, not only linear but also network-like processes with alternative execution can be supported by this concept. As shown in Figure 4 intermediary 1 may continue execution with intermediary 2a or intermediary 2b by choosing one of both for the further processing of the SOAP-envelope. This decision may be dependent on the evaluation of conditions in the business rules. For example, the execution may be continued with intermediary 2a if a condition is met, otherwise it is continued with intermediary 2b.

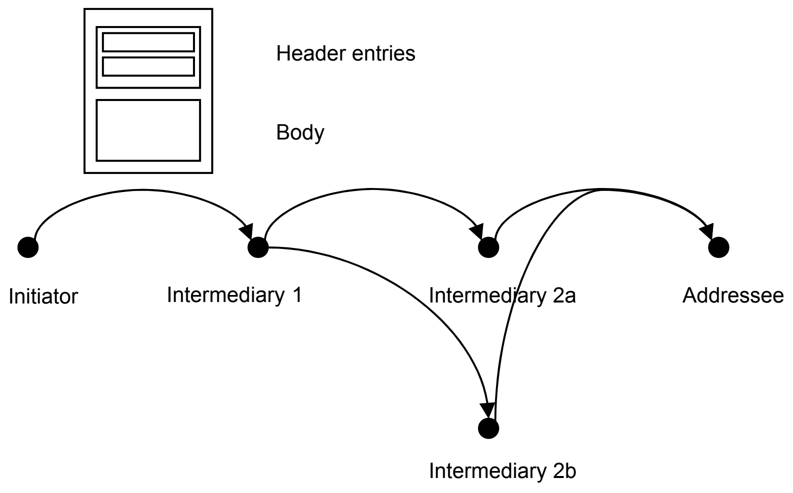

Figure 4: Alternative execution

Furthermore, for differentiating individual instances of a business process we need a concept to store individual status information and instance specific data. For example, if we consider an insurance claim process, then the instance specific data represents the concrete claims number 1,2,3, ..n with different data such as claim value, date etc. In the execution model process proposed here, for each instance of the business process, a separate SOAP envelope is used. In figure 5, we see two envelopes, each representing two process instances with different status. The existence of multiple envelope instances allows representing the different status of the process instances.

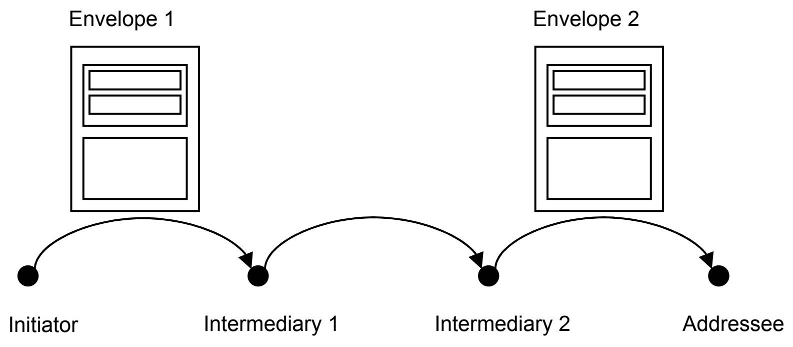

Figure 5: Instance representation using multiple envelopes

The execution of the business rules is done fully distributed, because no dependencies exist between the executions of business rules by different intermediaries. This is facilitated by the message-oriented processing of the business rules, which allows, that each business rule is executed independently by an intermediary. No interaction with other intermediaries is necessary except when receiving and sending of SOAP envelopes. This contrasts to RPC-oriented approaches, where a centralized

mechanism is necessary for initiating the – possibly distributed – processing of business rules.
The execution model proposed offers also a high degree of flexibility, because changes to the business rules can be easily implemented by changing the XML-document. Therefore, business rule changes require only changes to a text document. Other approaches such as the Meteor- and Meteor2-Project [Wang95], [MSKW96] require the recompilation of program code to implement changes to business rules.
Furthermore, the execution model proposed provides also the transparent evolution of business processes. Different versions of a business process may be represented by different versions of the SOAP envelope as shown in figure 6. Versioning of the SOAP envelope is done by using the XML namespace concept [XML]. By using different versions of the SOAP envelope, different versions of the business process may coexist transparently.

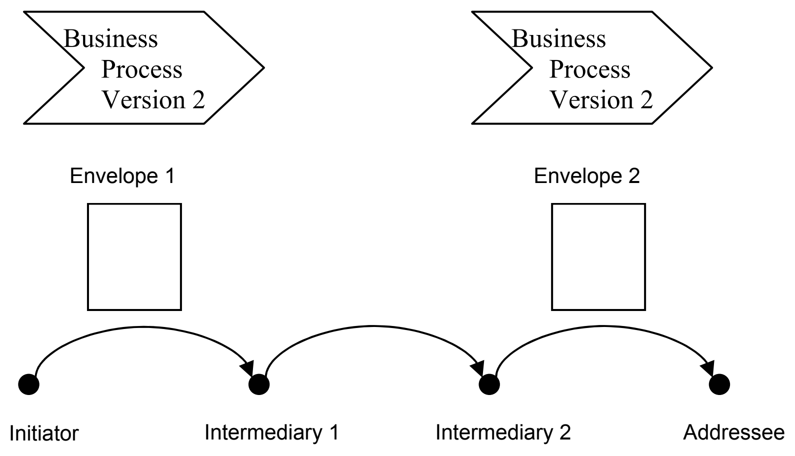

Figure 6: Instance representation using multiple envelopes

The evolution of business processes not only requires the change of business rules but also the integration of additional services. These additional services can be easily found by using UDDI (universal description discovery and integration) [UDDI], another web services technology. It provides a registry that allows quickly finding additional web services for implementing a process change. The integration is facilitated by using the web service description language WSDL [W3WS].
The capabilities of SOAP can be used to integrate services transparently into the business rules encoding the business process. The services may be scattered around the internet, implemented using different programming languages and running on different platforms, nevertheless they are transparently accessible by the use of web services. E.g. during a purchasing process, a check of the buyer’s solvency may be easily done by a web service based request to the issuer of the buyer’s credit card.

4. Related work

The distributed nature of business processes in e-business and e-commerce obstructs centralized approaches for process support such as workflow-engines, e.g. [WfMC]. Therefore, the support of business processes in distributed environments has been the subject of extensive research. The Broker/Services Model in combination with the EVE middleware [GeTo97], [GeTo98], provides the support of business processes with ECA-Rules. However, a centralized rule engine is needed. Also the WIDE-Project [WIDE], [CeGS97], [CGPP97] has the goal to support distributed business processes, but uses centralized rule engines. The Meteor- and Meteor2-Projekt [Wang95], [MSKW96] have the aim to create a fully distributed support of business processes by creating compiled task managers. However, the compilation process creates a huge effort to implement business process changes. The "Business Rules in e-Commerce"-Project [IBM2] focuses on semantic interoperability. The Business Rule Beans (BRBeans) Project [IBM3] still uses Enterprise Java Beans, a component-oriented middleware with similar restrictions as CORBA.

5. Summary and further work

The use of web services as foundation for the execution of business rules not only allows a fully distributed execution of business rules but also to cope with the integration of an evolving set of heterogeneous services and their independent evolution. Furthermore, using the namespace mechanism of XML allows seamlessly introducing different versions of business process instances, which may coexist harmoniously.

However, there still needs work to do. For example, concepts for storing the process state information have to be developed. They are needed to monitor the execution of business process instances. In addition, the model has to be extended by registry services for web services such as UDDI [UDDI].

References

[BoTW01] H. Boley, S. Tabet, G. Wagner: Design Rationale of RuleML: A Markup Language for Semantic Web Rules.

[CeGS97] S. Ceri, P. Grefen, G. Sánchez: WIDE - A Distributed Architecture for Workflow Management. RIDE97. Seventh International Workshop on Re-search Issues in Data Engineering, Birmingham, England, April 1997.

[CGPP97] F. Casati, P. Grefen, B. Pernici, G. Pozzi, G. Sánchez. WIDE Workflow model and architecture, http://dis.sema.es/projects/WIDE/Documents/ase30_4.ps.gz.

[Chap96] D. Chappell: Understanding ActiveX and OLE. Microsoft Press, Redmond, 1996.

[FeSi94] O. K. Ferstl, E. J. Sinz: From Business Process Modelling to the Specification of Distributed Business Application Systems - An Object-Oriented Approach. Bamberger Beiträge zur Wirtschaftsinformatik Nr. 20, Bamberg, 1994.

[FeSi96] O. K. Ferstl, E. J. Sinz: Flexible Organizations Through Object-oriented and Transaction-oriented Information Systems. Bamberger Beiträge zur Wirtschaftsinformatik Nr. 37, Bamberg, 1996.

[GeTo97] A. Geppert, D. Tombros: Ereignisgesteuerte Workflow-Ausführung in verteilten Umgebungen. In [JaBS97].

[GeTo98] A. Geppert, D. Tombros: Event-based Distributed Workflow Execution with EVE. Technical Report 96.05 (revised 1998). Department of Computer Science, University of Zürich, 1998.

[Giso01] D. Gisolfi: Web services architect, Part 3: Is Web services the reincarnation of CORBA?. July 2001. http://www-106.ibm.com/developerworks/library/ws-arc3/index.html

[GrSi02] S. Graham, S. Simeonov et. Al.: Building Web Services with Java, SAMS Publishing, Indianapolis, Indiana, USA, 2002

[HKMS94] Herbst, H., Knolmayer, G., Myrach, T., Schlesinger, M., The Specification of Business Rules: A Comparison of Selected Methodologies, in: A.A. Verrijn-Stuart, T. W. Olle (Eds.), Methods and Associated Tools for the Information System Life Cycle, Amsterdam et al.: Elsevier 1994, pp. 29 - 46 [IBM] www.ibm.com/developerworks

[IBM1] http://www-106.ibm.com/developerworks/library/ws-spmyths.html

[IBM2] http://www.research.ibm.com/rules/home.html

[IBM3] http://www.research.ibm.com/AEM/brb.html

[MSKW96] J. A. Miller, A. P. Sheth, K. J. Kochut, X. Wang: Corba-based run-time ar-chitectures for workflow management systems. Journal of Database Management, Special Issue on Multidatabases, Juli 1996.

[OMG] http://www.omg.org

[rule] http://www.dfki.uni-kl.de/ruleml/

[Sche94] A.-W. Scheer: Wirtschaftsinformatik. Referenzmodelle für industrielle Geschäftsprozesse, 4. Auflage. Springer Verlag, Berlin, 1994.

[UDDI] www.uddi.org

[W3] www.w3.org

[W3WS] http://www.w3.org/2002/ws/

[Wang95] X. Wang: Implementation and Evaluation of CORBA-Based Centralized Workflow Schedulers. Master's thesis. University of Georgia, August 1995.

[W3C] www.w3c.org

[WfMC] Workflow Management Coalition: http://www.aiai.ed.ac.uk/project/wfmc/

[WIDE] WIDE Projekt: Project Description and Objectives. http://www.sema.es/projects/WIDE/Objectives.html

[WSI] www.ws-i.org

[XML] www.xml.org