Composing Web Services Specifications: Experiences in Implementing Policy-driven Transactional Processes

Stefan Tai

IBM T.J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532, USA
stai@us.ibm.com

Abstract: The Web Services architecture defines various specifications that applications may wish to use in combination. In this paper, we investigate the composition of the Web services specifications for business process execution (BPEL) and transactional coordination (WS-Coordination). We report on our experience in implementing a policy-driven model to declaratively program transactional processes and discuss challenges in middleware integration to support this model.

1 Introduction

The Web services set of specifications define an XML-based platform for service-oriented computing [Al04]. These (proposed) open standards and the pervasiveness of Internet technologies together provide a foundation for the description, discovery, and interoperability of diverse applications in a loosely-coupled environment.

A variety of Web services specifications has been developed. Each specification addresses a specific concern and has been designed with composition in mind. The objective is that a subset of specifications can be selected and composed for use by an application, depending on application and interoperability needs.

The combined usage of some of these specifications is well-understood, such as WSDL for service description, SOAP bindings in WSDL for interaction, and UDDI registries holding WSDL descriptions for service discovery [Kh04]. However, this is not the case for all compositions of specifications, in particular those where different middleware systems implementing the individual specifications must be integrated. This includes compositions with the security, reliable messaging, and transactions specifications, which typically each require corresponding middleware support.

In this paper, we investigate this problem for the specifications for Web service composition and (transactional) Web service coordination: BPEL and WS-Coordination. BPEL is a language for creating service compositions in the form of business processes. WS-Coordination is a framework for context-based coordination of distributed activities, as required by distributed transactions. Both specifications require middleware system support, such as a BPEL process execution engine and a Web services transaction monitor.

We describe a model for composing BPEL and WS-Coordination that uses declarative policy assertions. Policies are used to extend standard BPEL definitions with coordination semantics, and they are further used to drive and configure the corresponding middleware integration. We discuss our experiences in prototyping this model.

2 Motivating Example

Consider a federated order processing and vendor managed inventory system such as the one introduced in [Fe03]. The system is used by car dealers to order parts from an automobile manufacturer; the manufacturer in turn obtains parts from a supplier operating multiple warehouses. All application communications in the system are built using Web services protocols. Here, we focus on the warehouse application that communicates with the supplier and other, subordinate warehouse services.

The warehouse application receives orders for parts from the supplier application. In order to tolerate potential message loss and/or temporary unavailability of the warehouse application, the supplier requires the use of a reliable messaging protocol. The protocol ensures delivery of messages sent; messaging middleware is used to (re-)send a message from the supplier to the warehouse until a response is received.

All incoming orders at the warehouse application are divided among a number of subordinate services representing physical warehouses and databases. For example, to ensure inventory coverage, 70% of an order may go to Warehouse 1 and the remaining 30% may go to Warehouse 2. An atomic transaction protocol is needed to ensure transactional semantics when updating the different databases. The warehouse application is the transaction client and the subordinate warehouse database services are transaction participants.

The warehouse application can be viewed as a (business) process that offers an interface for invocation as a service and that invokes other services as part of the process. The application comprises a sequence of activities including order receipt and order processing, some of which require the use of an interoperability protocol specification for reliable messaging or atomic transaction processing. In the following, we explore how such processes can be implemented by using and composing Web services specifications (BPEL, WS-Coordination).

3 Background: BPEL and WS-Coordination

The *Business Process Execution Language (BPEL)* is a choreography language for defining flows of a business process that composes various Web services [Th03]. Compositions are created by defining control semantics around a set of interactions with the services. The BPEL process itself, like any Web service, supports a set of WSDL interfaces so that it can be exposed and invoked as a regular Web service. This is illustrated in Figure 1(a). The interpretation and execution of BPEL processes requires a process execution runtime.

The *Web Services Coordination (WS-Coordination)* specification defines an extensible framework (independent from BPEL) which can be used to implement different coordination models that require a shared context [La04]. This includes traditional atomic transactions and long-running business transactions; interoperability protocols for these models based on WS-Coordination are defined in the *Web Services Atomic Transactions* and *Web Services Business Activity* specifications [La04]. Coordination middleware is needed for each coordination participant. This is illustrated in Figure 1(b).

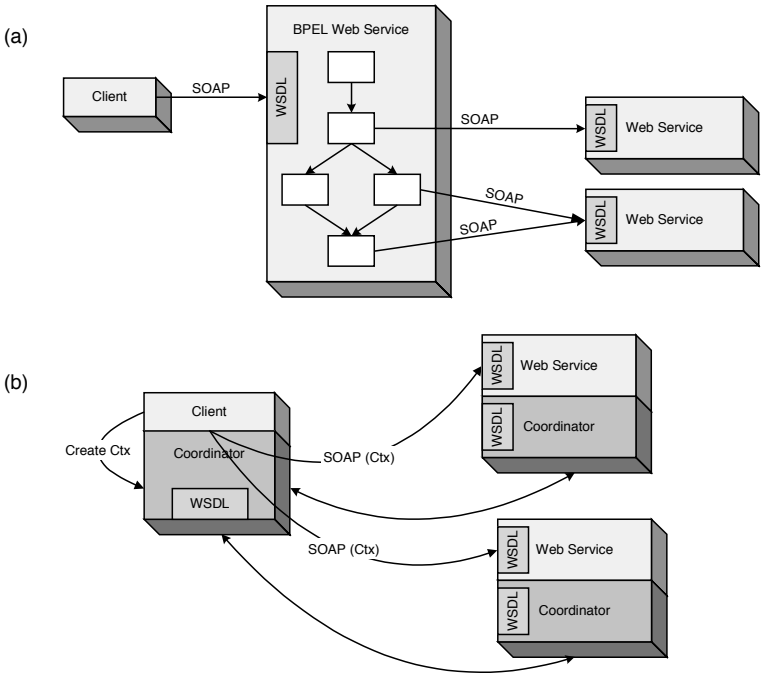

Figure 1: BPEL Web Services Composition (a) and Web Services Coordination (b)

Coordination middleware is used to create coordination contexts, to propagate contexts to participants, to register participants, and to process the messages of coordination protocols (such as the two-phase commit protocol for atomic transactions). The middleware implements the required coordination semantics, typically utilizing an existing data management system.

Both composition and coordination are essential techniques for creating service-oriented architectures. BPEL allows applications to be build by (internally) aggregating functionality of existing services, and WS-Coordination (and related protocol specifications) allows services to be (externally) coordinated to reach an agreed processing outcome.

In the warehouse application example described above, the warehouse application can be implemented as a BPEL process that appears as a regular Web service to the supplier, and that internally composes other, subordinate warehouse database services. The warehouse application also is the client of coordination middleware, coordinating the database services for transaction processing.

4 Objective: Combining BPEL and WS-Coordination

The combined use of BPEL and WS-Coordination can take different forms. A BPEL process, exposed as a regular Web service, can be coordinated (that is, the process is a participant in an externally created coordination). Or, the BPEL process coordinates a set of services (that is, the BPEL process is the coordination client).

In the first case, the BPEL process participates in the externally initiated coordination by accepting an incoming shared coordination context. The incoming context may be further propagated to the services that the BPEL process invokes, in which case the invoked services will register with and be enlisted as participants of the external transaction, too. This is illustrated in Figure 2(a).

If a registration and enlistment with a different (local, interposed) transaction coordinator is desired, however, the BPEL process may choose to not propagate the incoming context, but to create a new context. In this case, the BPEL process is a coordination client in addition to being a coordination participant. Interposed coordination is illustrated in Figure 2(b).

Whenever the BPEL process creates a coordination context – using a coordination middleware – for propagation to (a subset of) the services that it invokes, the process describes a *composition of coordinated services*. That is, in addition to business process control semantics, coordination control semantics are introduced.

In the warehouse application example, the warehouse process describes a specific case of Figure 2(b). The BPEL runtime executing the warehouse process is a coordination client, creating an atomic transaction context for coordination of its database services. The BPEL process however is not a participant in an external transaction. The process is a service that is invoked using a reliable messaging protocol.

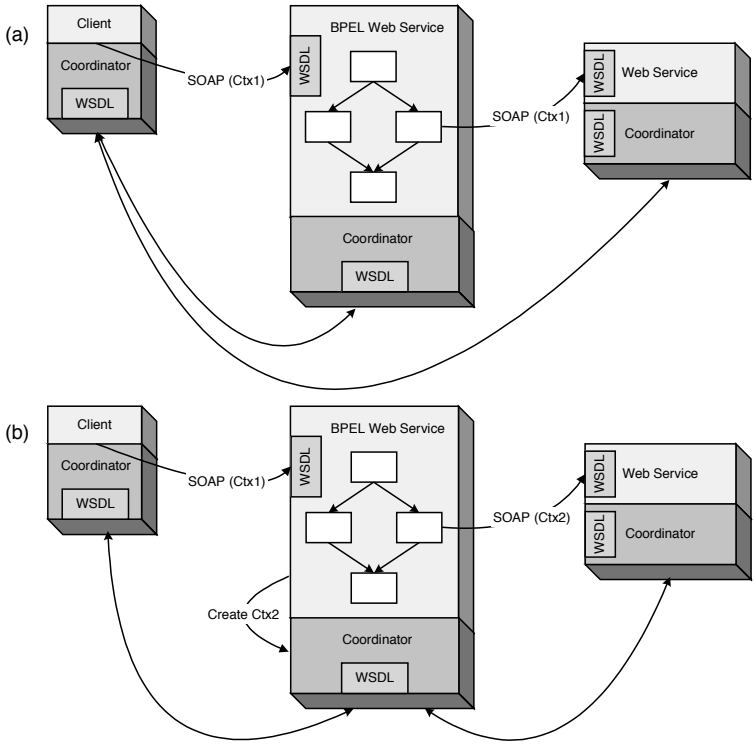

Figure 2: Combining BPEL and WS-Coordination (a) without and (b) with interposed coordination

In general, coordination requirements can vary both within a single BPEL process and between different processes as different transaction models (coordination types and protocols) are available and different transactional patterns can be implemented. In the warehouse application, there is only the need for a single atomic scope. However, other applications may require different process scopes to be coordinated using different coordination types. Or, rather than coordinating different services, all interactions with a single service partner may need to be coordinated as a transaction.

A high degree of flexibility is desired when extending BPEL with coordination semantics in order to align with the dynamic nature of service-oriented architectures: desired or required transaction models for interaction with service partners may only be determined late at runtime or may even change during process execution time. In the warehouse application example, the subordinate database services are known at deployment time. In other scenarios, services may be discovered at runtime and their (transaction, security, or other) interoperability requirements may only be known during process execution.

The composition of coordinated services accordingly requires paying careful attention to the (integration of the) underlying middleware systems: the BPEL process engine and the coordination middleware. The flexibility desired for selecting and varying coordination models requires the middleware systems to support dynamic transaction configuration.

5 Policy-driven Transactional Processes

We argue for a policy-based approach to non-intrusively attach coordination capabilities to BPEL process definitions, and to use policies to drive and configure the middleware systems.

5.1 Coordination Policies and Policy Attachments

We define coordination policies as declarative assertions of coordination behavior. A coordination policy is represented as an XML element that references the XML namespace URI of a published WS-Coordination coordination type. Sample coordination types are defined in the WS-Atomic Transactions and WS-Business Activity specifications, and coordination policies referencing these types can easily be authored. Concrete examples of coordination policies that use the XML syntax defined in the WS-Policy framework [HK04] are given in [Ta04].

Coordination policies can be attached to diverse Web services definitions, including WSDL port types (for transaction participants) and BPEL process definitions (for transaction clients), by means of XML extensibility and referencing mechanisms. The WS-Policy Attachment specification defines such mechanisms [HK04].

For the composition of coordinated services as motivated above, we propose the attachment of policies to two BPEL constructs: *scopes* and *partner links*. A BPEL scope is the demarcation of a group of activities of the process. Scopes are the units of data, fault, and compensation handling in a BPEL process. A BPEL partner link is a typed connector along which a conversation with another party occurs. By attaching a coordination policy to a scope or a partner link, a coordination requirement on the services of the scope or the partner link is expressed.

5.2 Coordinated BPEL Scopes

A scope with an attached coordination policy requires all services composed by the scope to be coordinated according to the declared coordination type. That is, when entering the scope a shared coordination context is created by the BPEL process runtime, and the context is propagated to all services used in the scope by including it in the application-level messages. The context includes the addressing endpoint of the coordination middleware to be used for participant registration, and all services receiving the context use the coordinator endpoint specified for registration. The services that are invoked are required to support the coordination type for the scope.

In the warehouse application example, the required coordination of the database services can be modeled as a regular BPEL scope with an attached coordination policy asserting the WS-Atomic Transaction coordination type. The database services in turn declare their support for the WS-Atomic Transaction coordination type by attaching a coordination policy to their WSDL definition.

Before closing the scope, any completion protocols required by the coordination type (such as the two-phase commit protocol) are performed. If the outcome of the completion protocol requires recovery, the coordination middleware initiates recovery for all remote services. For example, rollback requests are sent in case of atomic transactions and compensation requests are sent in case of business activities. Local activities that do not involve coordinated partners are recovered using standard BPEL compensation handlers.

5.3 Coordinated BPEL Partner Links

A partner link with an attached coordination policy (within a regular BPEL scope) implements coordination for all activities with that single partner only. The coordination context is created before the first interaction with the partner, and the context is propagated along all subsequent interactions with the partner. Required completion and recovery protocols are executed for the partner (as described above for coordinated scopes) before closing the scope that encompasses the coordinated partner link. Coordinated partner links are valid only within or across regular BPEL scopes that do not have a coordination policy attached. Otherwise, the coordination policy attached to a scope dictates the effective policy for its partners.

Figure 3 illustrates the proposed model. A sample coordination policy declaring support for the WS-Atomic Transaction coordination type is shown in the upper right corner of the figure. Other coordination policies can be authored. Coordination policies are then attached to the WSDL of the Web services that can be coordinated. A coordination policy also is attached to a scope within the BPEL process (left side of the figure), thus coordinating the Web services that are being invoked as part of the scope. Independently of that coordination, a policy is attached to a partner link definition (right side of the figure), illustrating the coordination of multiple activities with a single partner service.

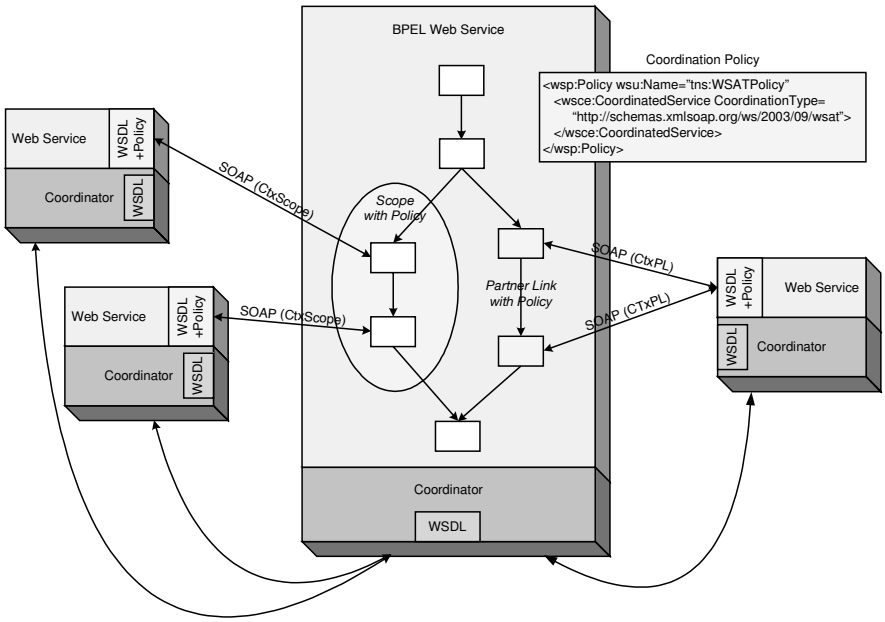

Figure 3: Using coordination policies and policy attachments to WSDL (for service providers) and to BPEL scopes and BPEL partner links

Our example of the warehouse application is illustrated in Figure 4. The application is implemented as a BPEL process that provides a WSDL interface for interaction with the supplier application. Both the warehouse application and the supplier application require a messaging middleware that supports reliable messaging [TMR03].

An atomic scope is defined as part of the warehouse application process using a coordination policy. The warehouse database services each declare support for atomic transactions, too. As described previously, the BPEL process execution runtime creates a coordination context when entering the scope, propagates the context to the database services using coordination middleware, and the database services in turn register with the warehouse application's coordinator. Completion protocols are driven between the coordinators before closing the scope.

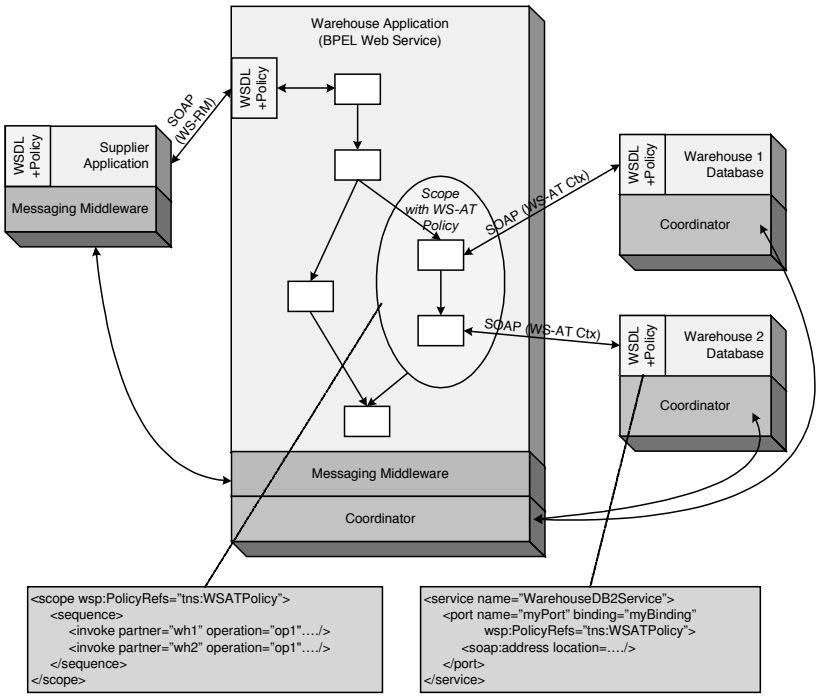

Figure 4: Warehouse Application Example

6 Middleware

The composition and coordination model described above enables an extensible variety of coordination types and protocols to be attached to a BPEL process definition, supporting two general transactional process patterns: *transactional scopes* and *transactional partner links*.

This declarative, simple programming model of using policies to complement the BPEL process definition, however, introduces additional responsibilities to the middleware layer. A policy middleware is required in addition to the BPEL process engine and the transaction monitor. The policy middleware supports the tasks of static process verification, distributed policy mediation and policy matchmaking, and policy-driven transaction configuration.

Static process verification is the task to ensure proper attachment and compliance of coordination policies to BPEL scopes and/or partner links. Distributed policy mediation is the task to request policy exchange and to negotiate policies with a partner. The partner's policies may only be available at runtime and policy mediation and a negotiation protocol are required to dynamically exchange and agree on a policy [Wo04]. The latter is the case when any one of the partners offers policy alternatives (such as different coordination types) among which a selection must be made. The selection is based on matchmaking criteria, a simple form of which is an exact match of names (such as the declared coordination type namespace).

Further, once policies are verified and selected for BPEL scopes and partner links, the policy middleware must communicate the policies to the BPEL process engine. The BPEL process engine in turn is a client of the transaction middleware: The engine creates coordination contexts and initiates the completion of coordinated activities. The transaction middleware is responsible for implementing the coordination models and driving the required coordination protocols.

Figure 5 illustrates the integration of these middleware systems. Static process verification (Step 1) is performed prior to process execution. Policy exchange (Step 2) and policy matchmaking and selection (Step 3) are performed at deployment time, and, if needed, during process execution time whenever a partner is dynamically bound. A selected policy becomes effective after it has been communicated to the BPEL process engine (Step 4). The BPEL engine may start one or more transactions in the course of process execution (Step 5) and invoke one or more service providers using application messages that carry a coordination context (Step 6). Each service provider receiving a context registers once with the coordinator (Step 7) for the appropriate transaction. Before process (scope) completion, coordination protocols are executed for the registered participants by the coordination middleware (Step 8).

7 Discussion

We implemented the model of policy-based composition of BPEL and WS-Coordination by integrating different middleware systems. The integration required the availability of appropriate APIs for the individual systems, which are not subject to standardization in the Web services set of specifications. In our experimental prototype, the WS-Policy compliant policy middleware, the BPEL process execution engine, and the WS-Coordination compliant transaction monitor all define proprietary (Java) interfaces for integration. It is not clear to what extent an integration of arbitrary middleware systems that support the Web services standards and are available on the market (such as existing BPEL runtimes and WS-Coordination compliant transaction services) is possible.

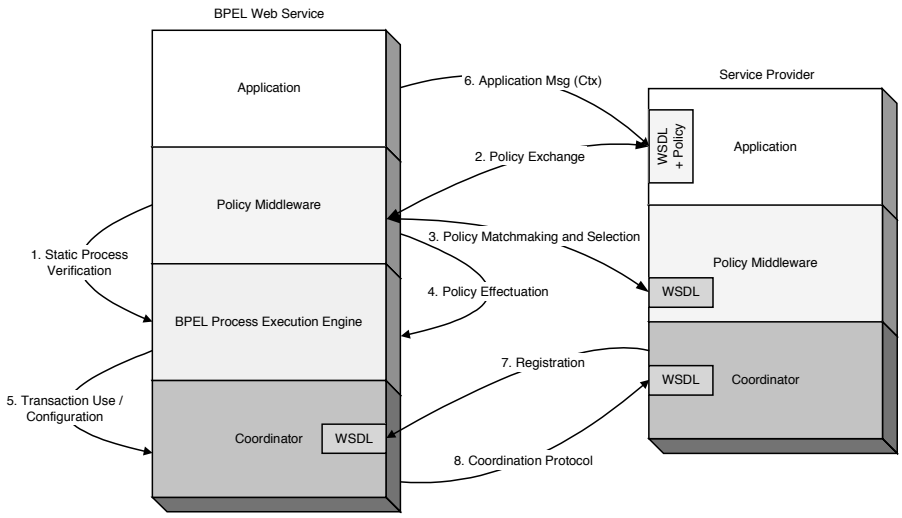

Figure 4: Integration of policy middleware, BPEL middleware, and coordination middleware and their remote interactions

The proprietary interfaces of middleware systems that we defined should ideally enable different compositions of Web services specifications. Middleware integration is not restricted to business process execution and transactional coordination, but is also required for Web services security, reliable messaging, and other interoperability concerns. However, not all compositions of specifications are needed or desirable. Further, the semantics of desirable specific compositions, and the integration of respective middleware, is subject of ongoing research. More experience in composing selected Web services specifications is needed to define and validate proprietary middleware APIs, as well as to validate the current Web services set of specifications.

While we believe policies to be a promising approach to compose Web services specifications, a number of open issues remain. Policies must be unambiguous and use well-defined vocabulary, and the information provided must be consistent and complete enough to allow for matchmaking (selection) and middleware configuration purposes. The WS-Policy framework provides a model and XML grammar to start with, but specific policies for interoperability concerns such as transactions, security, availability (and other, business-oriented issues) are yet to be better understood and agreed upon.

The declaration of more complex policy statements from simple policy assertions using aggregation and alternatives, both for a single concern and across concerns, must be carefully considered, as this may amplify the problem of middleware configuration. Ordering requirements may need to be introduced, for example, when combining concerns with security (encryption, authentication) and reliability (logging, persistence).

The policy-driven model for composing BPEL and WS-Coordination, as presented in this paper and in previous publications [TKM04], exemplifies the combined use of two published Web services specifications. Other coordination and composition models for Web services have been proposed, for example the WS-CAF [Bu03]. While these differ in some of the proposed features and technical details, the problem of (understanding) the combined use of separate Web services composition and coordination specifications remains.

Many workflow systems also support transactional coordination [LR04], and the transaction literature has proposed many ways to do (extended) transactions as well. These workflow systems typically implement proprietary solutions, but do not support dynamic integration of diverse coordination models based on open standards. Related work also exists with respect to attaching policies to definitions of business processes. In contrast to using policies as semantic annotations for purposes of service selection only [Be02], however, our work targets quality-of-service runtime interoperability and uses policies to configure required middleware.

The extent to which dynamic policy mediation and selection of coordination protocols is required remains to be seen. Current and future practice may show that pre-defined contractual agreements can alleviate the need for such highly dynamic architectures. The warehouse application presented in this paper is an example where dynamic policy mediation is not needed. On the other hand, varying quality-of-service requirements, the emergence of different (versions of) interoperability protocols, and the ability to adapt to changing system conditions (such as load or global security policies) are all increasingly critical factors in the selection and execution of Web services interactions. If services are discovered at process runtime, support for dynamic policy mediation and matchmaking is necessary.

8 Summary

The Web services set of specifications is emerging as a standard platform for service-oriented computing. A modular architecture comprising various lower-level and higher-level specifications is proposed, with the design objective for these specifications that they can be selectively and flexibly composed with each other. In this paper, we investigated this promise for the specifications for Web services composition and coordination: BPEL and WS-Coordination.

We proposed a policy-driven approach to extend BPEL definitions with coordination (protocol) behavior, and discussed the required middleware support for this model. While we are able to demonstrate that policies can be used to specify, associate and enforce coordination behavior for BPEL processes, to do so effectively required the integration of separate middleware systems. The modularity of the Web services specifications introduced the need for non-trivial middleware integration and configuration.

Acknowledgments

I am most grateful to my colleagues Thomas Mikalsen, Rania Khalaf, and Isabelle Rouvellou for many insightful discussions.

References

[Al04] Alonso, G. et.al.: Web Services. Springer-Verlag, 2004.

[Be02] Benatallah, B.; Dumas, M.; Maamar, Z.: Definition and Execution of Composite Web Services: The Self-Serv Project. In: Data Engineering Bulletin, 25(4): 47-52, 2002.

[Bu03] Bunting, D. et.al.: Web Services Composite Application Framework (WS-CAF). Published online at http://developers.sun.com/techtopics/webservices/wscaf/, 2003.

[Fe03] Ferguson, D. et.al.: Secure, Reliable, Transacted Web Services: Architecture and Composition. Published online at http://www-106.ibm.com/developerworks/webservices/library/ws-securtrans/, 2003.

[HK04] Hondo, M.; Kaler, C. (Eds.) et.al.: Web Services Policy Framework (WS-Policy). Published online at http://www-106.ibm.com/developerworks/library/specification/ws-polfram/, 2004.

[Kh04] Khalaf, R. et.al.: Understanding Web Services. In (Singh, M.P., Ed.): Practical Handbook of Internet Computing. CRC Press, 2004.

[La04] Langworthy, D. (Ed.) et.al.: Web Services Transactions specifications. Published online at http://www-106.ibm.com/developerworks/library/specification/ws-tx/, 2004.

[LR00] Leymann, F.; Roller, D.: Production Workflows. Prentice-Hall, 2000.

[Ta04] Tai, S. et.al.: Transaction Policies for Service-oriented Computing. In (van den Heuvel, W.-J.; Weigand, H., Eds.): Journal of Data and Knowledge Engineering 51 (2004): 59-79. Elsevier, 2004.

[Th03] Thatte, S. (Ed.) et.al.: Business Process Execution Language for Web Services Version 1.1. Published online at http://www-106.ibm.com/developerworks/library/ws-bpel/, 2003.

[TMR03] Tai, S.; Mikalsen, T.; Rouvellou, O.: Using Message-Oriented Middleware for Reliable Web Services Messaging. In (Bussler, C., (Eds.) et.al.): Proc. WES 2003. Springer LNCS 3095, 2004, p. 89-104.

[TKM04] Tai, S.; Khalaf, R.; Mikalsen, T.: Composition of Coordinated Web Services. In (Jacobsen, A., Ed.): Proc. 5th Int. Conf. on Distributed Systems Platforms (Middleware 2004), Toronto, Canada. Springer LNCS 3231, 2004, p. 294-310.

[Wo04] Wohlstadter, E. et.al.: GlueQoS: Middleware to Sweeten Quality-of-Service Policy Interactions. In (Estublier, J.; Rosenblum, D., Eds.): Proc. 26th Int. Conf. on Software Engineering (ICSE 2004), Edinburgh, Scotland, UK. IEEE, 2004, p. 189-199.