# Serviam Literature Survey

# Part IV

# Service-based Processes

2004-03-01

Jelena Zdravkovic

SERVIAM-LIT-04

Version 1.6

8 pages

## Table of Contents

8 pages

## 1 Service-Based Processes

To integrate business processes within or across enterprise boundaries it is not sufficient only to support simple interactions using standard messages and protocols. Business interactions require long-running activities that are steered by an explicit *process model* (Sharp, 2001). The process logic defines the sequence of *activities* and the routing directions that must be followed, as well as deadlines and other business rules being specified in the business process management system. For instance, processing a tourist trip, ordering materials from a supplier, managing a customer order, etc., are examples of business processes. Process activities may be simple (as sending a message), or complex, containing other activities. A process may execute within an organization or span across several of them.

As Web services are individual components limited in their capabilities, they need to be *composed* to create new, complex business interactions in the form of processes (Alonso, 2004). To bind Web services together a process model is needed to specify the order and rules by which operations are executed. In Web service environment, a process workflow is made up of activities that are implemented as operations that may span a single or multiple Web services. A *web services composition language* provides the means to specify such a process model. Figure 1 shows how service composition is positioned in the Web service technology stack:

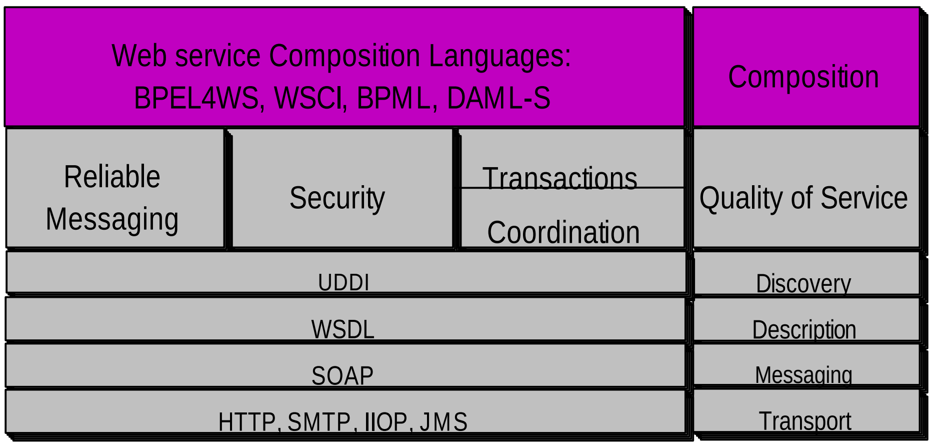

*Fig. 1. Web service technology stack.*

Process-based composition of Web services must be flexible, reliable and reusable to meet the changing needs of a business.

Flexibility can be achieved by providing a clear separation between the process logic and the web services used. This separation is achieved through service composition language engine. The engine handles the overall process flow, calling the appropriate Web services and determining the next steps to complete. With this approach, there are two possible scenarios for creation of service-based processes (Baresi, 2003): by first (*bottom-up*), a requested process is created out of set of available services; by second (*top-down*), the basic is a well-defined process with corresponding activities that have to be associated with suitable services. In addition, a unique service composition protocol is desired that can manage both EAI (in the form of *private processes*) and B2B (in the form of *public processes*) interactions involving Web services.

Reliability in service-based processes requires for strong support for advanced (long-running) transactional semantics and exception handling managing both internal and external errors.

Reusability in service composition may be achieved by composing existing processes as new, higher-level services. This is accomplished by exposing a service-based process with its own Web service interface so that other processes can then use it.

In service composition, two terms are distinguished. In processes that are controlled by a single party (such as most of intra-organizational processes), services are considered as *orchestrated* for process executions. In processes that are controlled by several parties (such as collaborative inter-organizational processes) in which each party involved describes the part they play in the interaction, the implemented services are *choreographed* (Peltz, 2003).

# 2 Web Service Composition Languages – Standards

## 2.1 General Aspects

Web service composition languages are designed to reduce the complexity required to orchestrate/choreograph web services, thereby increasing the overall efficiency of business processes. Without a standard, each organization is left to build its own set of proprietary business protocols, leaving little flexibility for Web services collaboration.

There exist several significant initiatives to describe how Web services can be composed into business processes: BPEL4WS, WSCI, BPML and DAML-S. BPEL4WS (Business Process Language for Web Services) specification (Leymann, 2003), proposed by IBM/Microsoft originates from IBM's Web Service Flow Language (WSFL) and Microsoft's language for business process design (XLANG). Sun, BEA, SAP, and Intalio have introduced another web services composition language: WSCI (Web Service Choreography Interface) (Austin, 2003). In contrast to these commercial, XML-based standards, researchers from several US universities are developing a unique Web service markup language called DAML-S (DAML-S, 2003), that among other features support composition of services. The Business Process Management Initiative organization (BPMI.org), which defines open specifications, has developed the BPML (Business Process Markup Language) (BPML, 2002). In addition, OASIS and UN/CEFACT support ebXML (Electronic Business using eXtensible Markup Language), a specification for B2B business protocol modeling (ebXML, 2004).

As there are many standards for service composition languages, it is useful to have measures for comparing them. A language should support design for any of business protocols and interactions that an organization may be involved in (EAI, B2B, etc.). In addition, language semantic should support not too complex process definitions. For the quality of process modeling, the languages may be compared by supported control-flow patterns (Aalst van der, 2004). Each of these patterns corresponds to a routing construct, often needed when designing a process.

## 2.2 BPEL4WS

BPEL4WS is, currently, one of the most perspective standards for Web service composition. BPELWS is meant to be used to model the behavior of both *executable* and *abstract* processes. Executable business processes model service orchestration, i.e. actual behavior of a participant in a business interaction. Abstract processes are descriptions of business protocols that specify visible message exchange behavior (service choreography) of each of the parties involved in the protocol, without revealing their internal behavior. The language has an XML-based grammar for describing the control logic required to coordinate web services participating in a process flow.

BPEL4WS supports separation of process and service logic, as it is essentially a layer on top of WSDL. WSDL defines the specific service operations and BPEL4WS defines how the operations can be sequenced. WSDL data types are used within a BPEL process to describe the information that passes between requests. Every BPEL process may be exposed as a web service using WSDL.

# 3 Transaction Management

## 3.1 General Aspects

Service-based processes require transactional support in order to bind loosely coupled services into cohesive units of work and guarantee consistent and reliable execution.

*The conventional transaction model* (Conolly, 1998) conforms to ACID rules (atomicity, consistency, isolation, durability). A transaction either commits or automatically rolls back cancelling performed operations. If a transaction is complex, results of sub-transactions are visible only to the parent transaction (*close nesting*) and resources are blocked as long as the main transaction does not commit (*two-phased commit*). The conventional model is, therefore, suitable for transactions that are tightly coupled and occur within a system or between several systems over short periods of time (such as database transactions).

In business processes, transactions run over loosely coupled activities of diverse nature (i.e. non-data-based, such as sending a letter, for example) and may have long duration (Jajojdia, 1997). In addition, to provide on-time process delivery, intermediate results have to be available to all participants and activities should be able to execute concurrently (i.e. asynchronously). Thus, *the business transaction model* has to be imposed in an extended, relaxed sense compared to the conventional specification. The relaxation of transactional semantics in the business model is reflected through:

- relaxed atomicity: a business transaction rolls back executing compensational counterparts of committed sub-transactions, in reverse order, until a consistent process state is reached (within the business transaction or on its beginning),
- relaxed isolation (*open nesting*): results of sub-transactions are globally visible.

In the service-based processes a way to design a reliable and efficient transactional framework is to combine conventional (so called *atomic*) and business transactions (Papazoglou, 2003), (Cabrera, 2002). Atomic transactions (AT) administer smaller and shorter interactions, while business transactions rule larger and more durable interactions. Atomic transactions are made up of services that all agree to enforce a common outcome (*commit* or *abort*) of the entire transaction. Atomic transactions must use the two-phase commit protocol and have a coordinator to manage the coordination protocol messages, such as prepare, commit, cancel, that are sent to the participating services within a given atomic transaction. Within an atomic transaction, the operations are exposed as a single transactional web service and the service may consist of processes with other services. Atomic web services usually represents core business processes, such as order entry, shipment of goods, etc. Business transactions (BT) aggregate atomic transactions, as shown in Figure 2.

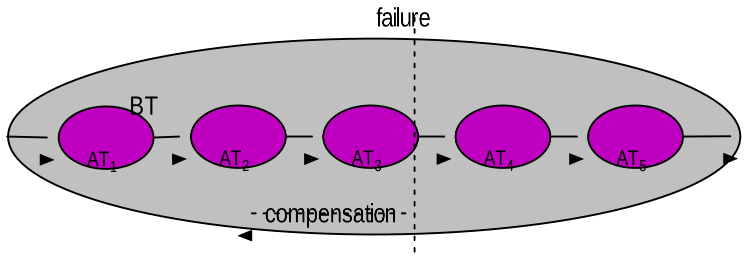

*Fig. 2. Two-level transactional model for service-based processes.*

Furthermore, each business transaction may be composed of other business transactions (i.e. arbitral nesting). Within a business transaction, each service (a sub-transaction) enforce outcome independently of other services, visible to everyone. When a service fails, the parent transaction rolls back to a consistent state by compensating services that had successfully completed.

### 3.2 BPEL4WS

In BPELWS, WS-Transaction and WS-Coordination specifications (Cabrera, 2002) complement the core language to provide mechanisms for transaction support and coordination of multiple Web services. WS-Transaction support both atomic and business transaction models, while Ws-Coordination is used to coordinate service executions in the both models.

## 4 Error-Handling

### 4.1 General Aspects

When executing a service-based process, *errors* might occur either in the services (*system failures*) being invoked or within the process itself (*semantic failures*) (Eder, 1996). System failures comprise information technology and application failures, which lead to an abnormal termination of a service. Semantic failures comprise exceptions within the process, which lead to an unsuccessful (but not abnormal) termination of an activity (for example, hotel reservation fails).

System failures cause inconsistent state of the process that must be first solved by the crashed service fault handler. After the service is recovered (cancelled or completed), the process will continue to execute or, if a consistent transaction state is not reached, process fault handlers will undo (compensate) previously completed activities (i.e. services).

Semantic failures require only process fault handlers to compensate already completed activities until a consistent process state is reached from where the process can continue.

### 4.2 BPEL4WS

BPEL4WS, for example (Khalaf, 2003), provides a mechanism to explicitly catch system errors (service level) and handle them by executing subroutines specified in *fault handler* elements. Additionally, *compensation handlers* allow defining certain activities that will be invoked to compensate activities that have completed and have to be undone due to a semantic error (process level). In distributed processes, spanning several parties and systems, WS-Transaction and WS-Coordination are used to control and coordinate execution of both fault and compensation handlers.

## 5 Summary

Web services are individual components, and therefore, they need to be composed in the form of processes to create new, complex business interactions. Process models are needed to specify the order and rules by which web services are executed. A web services composition language provides the means to specify such a process model.

Web service composition languages are designed to reduce the complexity required to compose web services, thereby increasing the overall efficiency of business processes. Without a standard, each organization is left to build its own set of proprietary business protocols, leaving little flexibility for Web services collaboration. There exist several significant initiatives to describe how Web services can be composed into business processes: BPEL4WS, WSCI, BPML and DAML-S.

Service-based processes require transactional support in order to bind loosely coupled services into cohesive units of work and guarantee consistent and reliable execution. The conventional ACID model is suitable only for transactions that are tightly coupled and occur within a system or between several systems over short periods of time (such as database transactions). In business processes, transactions run over loosely coupled activities of diverse nature (i.e. non-data-based, such as sending a letter, for example) and may have long duration. The business transaction model relaxes the conventional requirements for atomicity and isolation – activities commit globally and, when necessary, they are roll-backed by compensations. In the service-based processes a way to design a reliable and efficient transactional framework is to combine conventional (so called *atomic*) and business transactions. Atomic transactions (AT) administer smaller and shorter interactions, while business transactions rule larger and more durable interactions.

In service-based processes errors are handled on two levels: service (system failures) and process (semantic failures). System failures comprise information technology and application failures, which lead to an abnormal termination of a service. The failures cause inconsistent state of the process that must be first solved by the crashed service fault handler. After the service is recovered, the process will continue to execute or, if a consistent transaction state is not reached, process fault handlers will compensate previously completed services. Semantic failures comprise exceptions within the process, which lead to an unsuccessful termination of an activity. These types of failures require only process fault handlers to compensate already completed activities until a consistent process state is reached from where the process can continue.

## 6 Summarized Sources

In this section, short summaries of resources that discuss the important aspects of service-based modeling presented. These sources have been selected for inclusion here because they contain important insights, and/or they document the current state of art regarding the composition of web services. These sources, and others, have been used as an input to the overview. Other sources used for the overview are listed in the references section.

**Alonso G., Casati F., Kuno H., Machiraju V.: ”Web Services ”: Springer-Verlag (2004)**

In this book, based on their academic and industrial experience, Alonso and his co-authors clarify the fundamental concepts behind web services and explain why web services present the natural evolution of conventional middleware necessary to meet the challenges of enterprise and B2B application integration. In the context, the authors discuss in details existing standards, solutions and future challenges.

**Leymann F. et al.: “Business Process Execution for Web Services – Version 1.1”**: IBM Developer Library (2003). ftp://www6.software.ibm.com/software/developer/library/ws-bpel11.pdf

This paper is the specification for BPEL4WS. It is very useful, as it clarifies in detail one of the leading language standards for composition of web services as well as some of the core concepts behind service-based process modeling.

**Papazoglou M.: ”Web Services and Business Transactions”**: In World Wide Web: Internet and Web Information Systems 6(1):49-91, March 2003

The paper describes in detail a two-layered transactional framework for service-based processes.

**Eder J., Liebhart W.: “Workflow Recovery”**: Proceedings of the 1th IFCIS International Conference on Cooperative Information Systems (1996)

The paper describes the type of errors in workflow based processes and explain mechanisms for their handling.

## 7 References

### 7.1 Books

Connolly T. et al.: “Database Systems”: Addison –Wesley Longman Limited (1998).

Jajodia S., Kerschberg L.: “Advanced Transaction Models and Architectures”: Kluwer Academic Publishers, Norwell, Massachusetts (1997).

Sharp A., McDermott P.: “Workflow Modeling”: Artech House, Boston (2001).

## 7.2 Articles

Aalst van der W.M.P Home Page: http://www.tm.tue.nl/it/research/patterns, Accessed 17 January 2004.

Austin D., et al.: “Web Service Choreography Requirements 1.0”: W3C Working Draft (2003). http://www.w3.org/TR/2003/WD-ws-chor-reqs-20030812/, Accessed 17 January 2004.

Baresi L., Matera M., Plebani P.: ”Models and Technologies for e-Service Composition”: The First International Conference on Service Oriented Computing (ICSOC03), Trento, Italy, 15-18 December (2003). http://www.eusoc.net/, Accessed 02 February 2004.

Business Process Management Initiative: “Business Process Modeling Language”: www.bpmi.org (2004), Accessed 22 January 2004.

Cabrera F., et al.: “Web Service Transaction Specification”: IBM Debveloper Library (2002). http://www-106.ibm.com/developerworks/library/ws-transpec/, Accessed 22 January 2004.

“DARPA Agent Markup Language Services (DAML-S) Version 0.9” (2003). http://daml.semanticweb.org/services/daml-s/0.9/, Accessed 16 January 2004.

”ebXML Technical Specifications”: www.ebXML.org, Accessed 18 January 2004.

Khalaf et al: “BPEL4WS – Correlation, Fault Handling and Compensation”: IBM Developer Library (2003). http://www-106.ibm.com/developerworks/webservices/library/ws-bpelcol6/, Accessed 02 February 2004.

Peltz C.: “Web Service Orchestration: a Review of Emerging Technologies, Tools and Standards”: HP Tec.Report. http://devresource.hp.com/drc/technical_white_papers/WS-Orch/WS-Orchestration.pdf, Accessed 16 January 2004.