Sub-project Architecture

Service Orchestration

- Applying Executable Processes

2004-07-01

Martin Henkel, Jelena Zdravkovic

SERVIAM-ARC-11

Version 0.9

12 pages

Table of Contents

1 Introduction

As the number of Web services that an organisation requires increases, the importance of service coordination becomes apparent. For example, having a large amount of services, all exchanging messages, make it necessary to define rules for message routing, message translation and message sequencing. A set of upcoming technologies loosely called “executable process languages” aims to solve the coordination problems of Web services.

The purpose of this document is to give a brief description of executable processes, their basic concepts and which kind of problems they solve.

2 Processes – Basic Concepts

Executable processes are built on the fundamental concepts of activities, control flow and events:

- *Activities* represent some form of work done by a system, for example a calculation, or storing information in a database.
- *Control flow* is the dependencies of execution between the activities, for example the control flow describes if an activity should be performed before or after another activity.
- *Events* are triggers that set of when something happens, for example when a certain time has passed, or a system sends a message. Events can trigger the execution of activities.

Figure 1 depicts a basic process. The figure is drawn using the Business Process Modelling Notation, BPMN (White, 2004). Start and end events are depicted by circles, while arrows depict the control flow between activities.

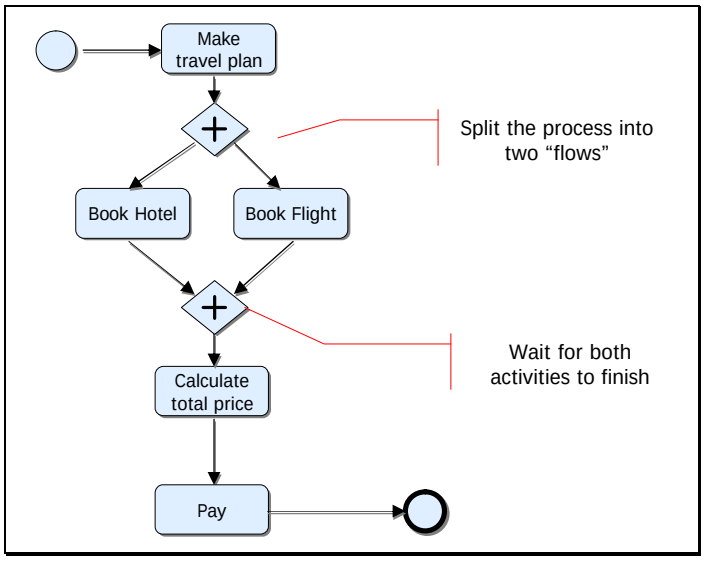

Figure 1, A very simple example of the use of process concepts.

Activities, event and control flow form the basis for describing processes, but an executable process must also contion descriptions of messages, transactions and roles. These other aspects must also be designed if it should be possible to coordinate web services with process

descriptions. The next section describes how processes can be related to web services, then the next chapter deals with aspects of process design.

2.1 Web Services and Processes

By combining process descriptions (such as the one in figure 1) with web services it is possible to describe how a set of web services shall interact. A process can be related to services in the following way:

a) A process can implement a service, that is, the process description depicts the inner workings of a service.

b) Activities in a process can call operations on a service.

c) Services can trigger events in the process by sending messages.

The three "connections" between services and processes can be depicted in a graphical process description by introducing the concept of partners or "services" and the concept of message passing. Figure 2 shows a process with services depicted as rectangles encompassing activities within the service, and message flows.

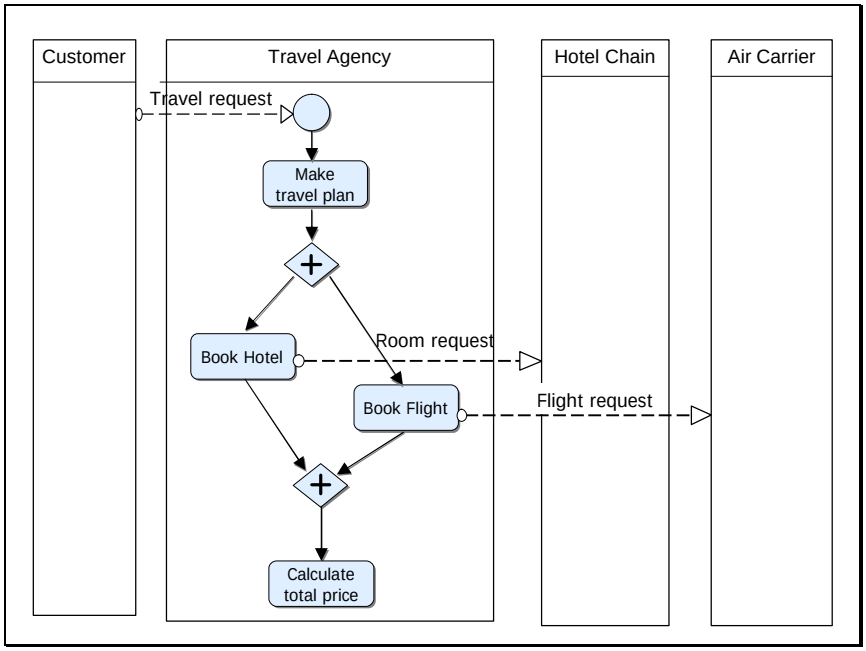

Figure 2, Simple process with services and message exchanges.

The figure above illustrates the three ways to relate a service to a process. The travel agency process is implemented (a) by the Travel Agency service. Some of the activities in the process utilises (b) the Hotel chain service and the Air Carrier service by sending messages to them. The customer service triggers an event (c) in the process by sending a Travel Request message.

Note that the relation between processes and services is quite simple, it is the same relation as between the implementation of a method (process) and a class interface (service). However, when depicting business processes the relation to web services in not quite as simple. We will discuss business processes in chapter 4.

3 Aspects of Process Descriptions

As seen in the previous chapter, processes that interact with services can not only be described using activities, control flow and events. In this chapter we examine the aspects of services that need to be considered when design an executable process. This chapter is an extension of the description presented in (Henkel and Zdravkovic, 2004).

As described in (Jablonski, 1998) and (Rausch-Scott, 1997) processes can be described by five main aspects functional, behavioural, organisational and transactional.

The basic aspect of process specification is *functional*. The functional perspective describes how a process is decomposed, i.e. what activities are to be executed. Functionality of an activity is depicted by name, which uniquely identifies the goal of the activity; by message exchange, designated with input and/or output documents of the activity; and by constraints, which depict activity-internal constraints, such as pre-conditions and post-conditions. The functional aspect, thus, depicts only the semantics of activities. Implementation of activities, however, is not part of the process specification. The result of designing the functional aspect of a process is a set of named activities and their post- and pre-conditions.

The *Behavioural* aspect depicts process control flow, i.e. when an activity is to be executed in relation to others. For specification of dependencies and coordination rules among activities, process specifications rely on a set of basic control flow constructs: sequence, parallel execution (AND split), synchronization (AND join) and conditional branching (OR/XOR split/join). The result of designing the behavioural aspect of a process is the sequencing of activities by the use of control flow constructs.

The *informational* aspect concerns process data. In a process specification, data are information concepts that are used as process attributes upon which flow rules are set and controlled, as well as information that the process exchanges with the external environment. An information concept is depicted by content and structure of contained data. The result of designing the informational aspect of a process is a set of message structures, and information structures that the process use internally (for example to store temporary process states).

The *organizational* aspect depicts the distribution of responsibility of executing the activities. In order to model business of an organization, personal and technical resources have to be mapped to specific roles. Roles are related to activities they are intended (allowed) to execute. By using roles it is possible to dedicate and control responsibilities of parties engaged in a process. The result of designing the organisational aspects of a process is a set of roles, each role is responsible for executing a set of activities. For example in figure 2, the rectangles “Customer”, “Travel Agency”, “Hotel Chain” and “Air Carrier” all represent roles.

The *transactional* aspect concerns consistent execution and recovery of a set of activities. The design of the transactional aspect includes defining what to be considered as consistent states of the process, thus depicting particular transactional boundaries (scopes). In addition, process transactions may comply to different models, as they contain loosely coupled activities that may have short or long duration. The atomic transaction model is used to administer a set of shorter activities, that all agree to enforce a common outcome by two-phase commit. In contrast, business (long-running) transaction model rules more durable activities, where each activity enforces a globally visible outcome independently of the others; when an activity fails, the parent transaction rolls back to a consistent process state by compensating activities that had successfully completed. The result of designing the transactional aspects of a process is a selected transaction model (long-running or atomic) and a grouping of activities into transaction boundaries.

Table 1, below, summarises the five aspects of processes.

Aspect	Design questions	Result of aspect design
Functional	▪ How do we break down the functionality into a set of activities? ▪ How much functionality should an activity encompass? Select granularity.	▪ Activities and events.
Behavioural	▪ Could activities be executed in parallel? ▪ Must the activities be executed in a certain order? ▪ Should certain activities only be executed under certain conditions?	▪ Control flow dependencies between activities. ▪ Control flow constructs (split, join, conditional statements).
Informational	▪ What are the contents of the messages that the process should produce? ▪ Does the process need any internal information to hold process state?	▪ Messages structures ▪ Structure of process-internal information.
Organisational	▪ Who/what will be responsible for executing activities? ▪ Who/what can send messages to the process? ▪ Who/what can receive messages from the process?	▪ Roles/Services.
Transactional	▪ What kind of transaction support is needed? ▪ Which activities should be executed within a transaction?	▪ Transaction model. ▪ Transaction boundaries.

Table 1, Summary of the five process aspects.

In addition to the above defined aspects, by the model of Jablonski and Rausch-Scott, processes have to cope with other aspects, such as causal and historical (data logging). Those concepts are not, however, used in process specifications; they are rather related to the execution environment of the process.

4 Applicability

Process descriptions can be applied in several ways. Firstly, graphical presentations of processes are excellent for designing and documenting the dynamic aspect of businesses and software systems. Secondly, if all five aspects of the process are designed, the process can be executed in a system.

Graphical presentation of processes has several advantages that make them interesting to use:

- Graphical process representations are often easy to understand for business people.
- Graphical representations of processes enable easy monitoring of process execution.

The first point is only applicable if the process is designed such that it is possible for business people to understand it. Including to much technical details in the process will make the process impossible to understand, even thought it is represented in graphical form.

If the process is designed such that it is possible to execute it, the process description can govern the sometimes advanced execution flow of activities. The process can be implemented in any language (like java or C++), however, there exist process languages that are specially designed for process execution. The Business Process Execution Language for Web Services

(BPEL4WS) is a XML-based language especially designed to be used together with Web services (Leymann, 2003).

Using specific process execution languages, instead of ordinary programming languages, have three major advantages:

- Process languages contain special constructs for control flow control, enabling easy parallel execution and synchronization of activities.
- Process languages are designed to handle multiple process instances (this is called "correlation").
- Specially adapted middleware products (such as Microsoft Biztalk) make it easy to handle processes in a standardised way. This makes is possible for the middleware to support advanced features like transaction handling and fail-over for processes.

Note that the first point is more advanced than it seems at first hand. Programming parallel execution in ordinary programming languages normally means that several execution threads have to be used. This implies that thread synchronization must be implemented, which for advanced processes can prove to be difficult. Another issue is that middleware servers such as those based on Enterprise Java Beans (EJB) does not allow the use of threads at all.

4.1 Types of Executable Processes

Executable process descriptions can be applied in numerous ways. However, two different areas of application can be recognised. Firstly, processes can be designed to close resemble the activities in a business, these processes are simply called "business processes". Secondly executable processes can be used/designed to solve technical integration issues, such as handling message routing. We call these processes "Technical processes". Note that both types of processes can utilise the advantages of process execution languages and graphical presentations previously pointed out.

Technical and Business processes – design differences

The focus of the design of technical and business processes is different. The following sections describe the differences in the design of technical and business processes. The description is an extension of the discussion presented in (Henkel and Zdravkovic, 2004).

When designing *business processes*, the focus is to "automate the business", i.e. to solve problems that are expressed in business terms. Thus, the aim of the design is to model and later on to implement business processes that capture the message exchanges, activities and roles that are part of a business. For instance, parallel activities in a process will be utilized to denote concurrent business operations. Another example is that a sequence of activities with specific messages to exchange will be used to steer business behaviour according to business contracts. Such processes have the advantage of being easy to understand for people within the business. Another advantage is that the close mapping between the business and the process makes it possible to easily rearrange the activities and sequences when the business changes. Workflows are an example of systems that closely depict the business process as understood by people engaged in the business.

When designing *technical processes*, the focus is to leverage business support by utilizing existing software services. This requires for solving system integration as well as synchronization problems. Just as with modelling processes from the business perspective, the final goal is to support the message exchanges and activities of the business. However, this common goal is reached by integration of existing services. Thus, when constructing technical processes the designer must deal with both business behaviour and the behaviour of existing services. The modelled process will at least partly describe the behaviour of the existing system. For example, it

might be decided that parallel activities must be used because the same information is required to be sent to two back-end services. Another example is that a set of alternative activities will be used to enable choice of a system used for communicating a message. These concepts are considerable more "technically" oriented compared to those dealt with when designing pure business processes. Middleware solutions that utilise integration patterns such as message routing, splitting and joining to interconnect back-end systems are an example of where process descriptions partly deal with system issues rather that business issues. By designing processes from the technical perspective it is possible to coordinate systems interactions in a straightforward way.

Design Summary

Table 2, below, compares the design of technical and business process using the five process aspects described in chapter 3.

	Functional	Behavioural	Informational	Organisational	Transactional
Design task	Decompose into activities	Sequencing of activities	Structure message and process information	Assign responsibility of process execution to roles	Select transaction model and transaction boundaries
Business process design	Decompose according to business activities in the organisations	Governed by business rules and contract	According to business concepts	Business partners, roles	Keep consistent business states
Technical process design	Decompose according to operations in existing services	Governed (partly) by features of the existing systems	Business concepts split according to service needs. Additional system specific information.	Systems and services	Keep systems and data in a consistent state

Table 2, Design of technical and business processes compared.

5 Example Case

In this chapter we will look closer at a "process case" supplied by Sandvik. This case describes an order-scenario where incoming order request should be verified, and then sent to a back-end ERP system. The intension of describing this case is to show an example where an executable process can be applied to coordinate the message exchanges between systems.

We start examining this case by drawing a graphical process model, we then examine how this model can be implemented using the process language BPEL4WS.

5.1 Graphical Process Model

When starting to analyse a process it is often god to start analysing the functional, behavioural and organizational aspects. Figure 3, below, shows the order process, its event and activities (functional aspect), the dependencies between activities (behavioural aspect) and the parties involved (organisational aspect).

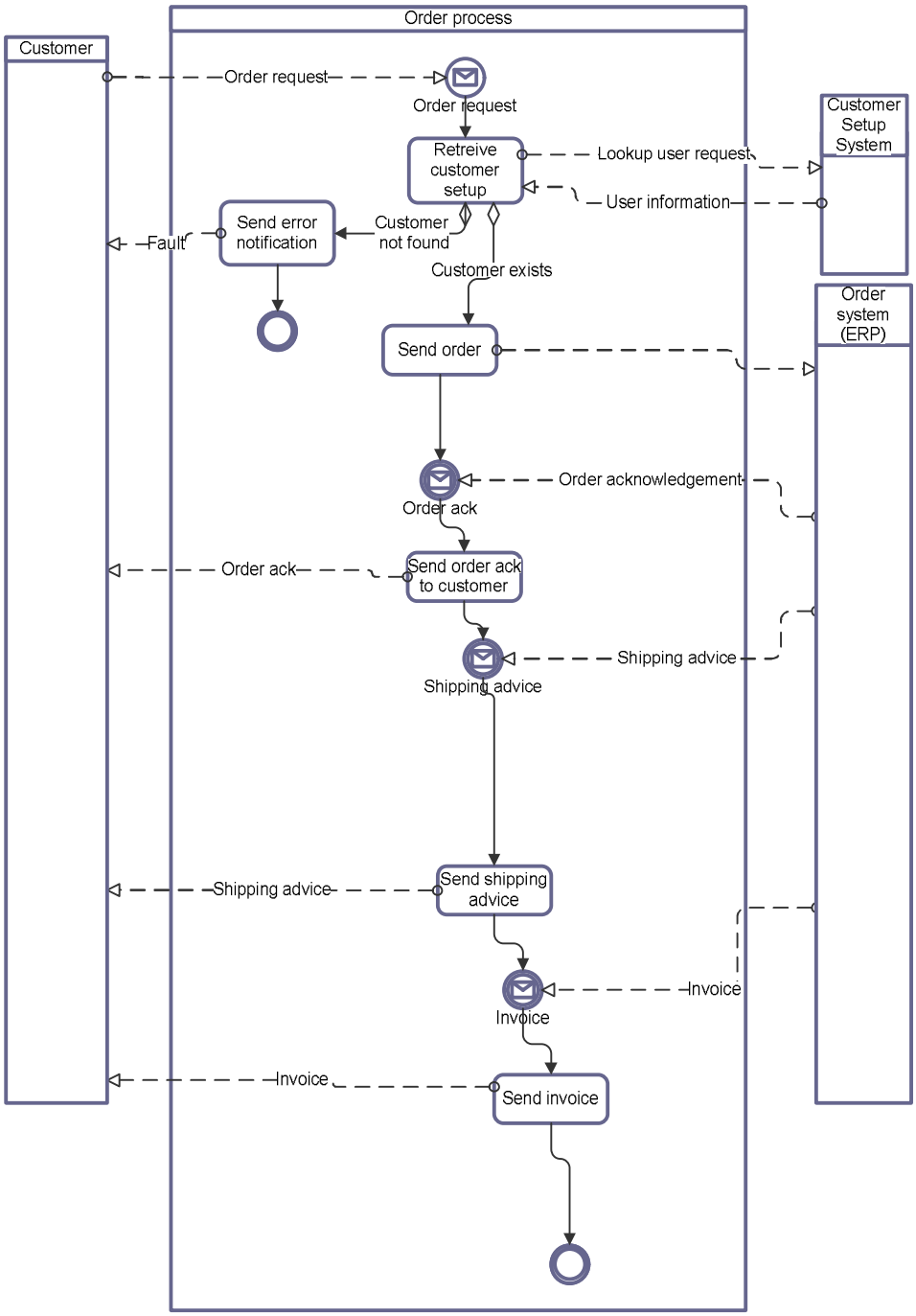

Figure 3, The order process drawn in BPMN

The process contains the following main steps:

1) An incoming order request from a customer triggers the start event in the process (the start message event is shown as an envelope).
2) The process retrieves customer information from the Customer Setup System. The customer setup contains the customers preferred ways of communication.
3) If the customer does not exist, a fault is returned to the customer and the process ends.
4) The order is sent to the back-end ERP system for processing.

5) When the process receives order acknowledgement, shipment advice, and the invoice from the ERP system, the customer is notified. Note that the message-event symbol (the encircled envelope) means that the process should halt until it receives a message.

In this scenario it is interesting to note that the process does not always "own" the control flow. For example there are several message-events that make the process wait to receive information from the ERP system. This means that the ERP system decides when the process can continue.

Even thought the described process is simple, we can achieve several advantages in implementing the process as a separate entity:

- Having a separate process (instead of incorporating it into the ERP system) makes it easy to combine system resources (in this case the Customer Setup System and the ERP system).
- It is easy to adjust the protocol for communication with the customer by changing the process implementation. In the actual case at Sandvik, both FTP and HTTP where possible communication protocols.
- A separate process description enables tracking on the process level. This is, the status of an order can be decided by looking at the state of the order process.

In the next section the model will be evolved by adding (a part of) the informational and transactional aspects as well.

5.2 Adding more aspects to the model

Figure 4, below, includes the transactional and informational aspects.

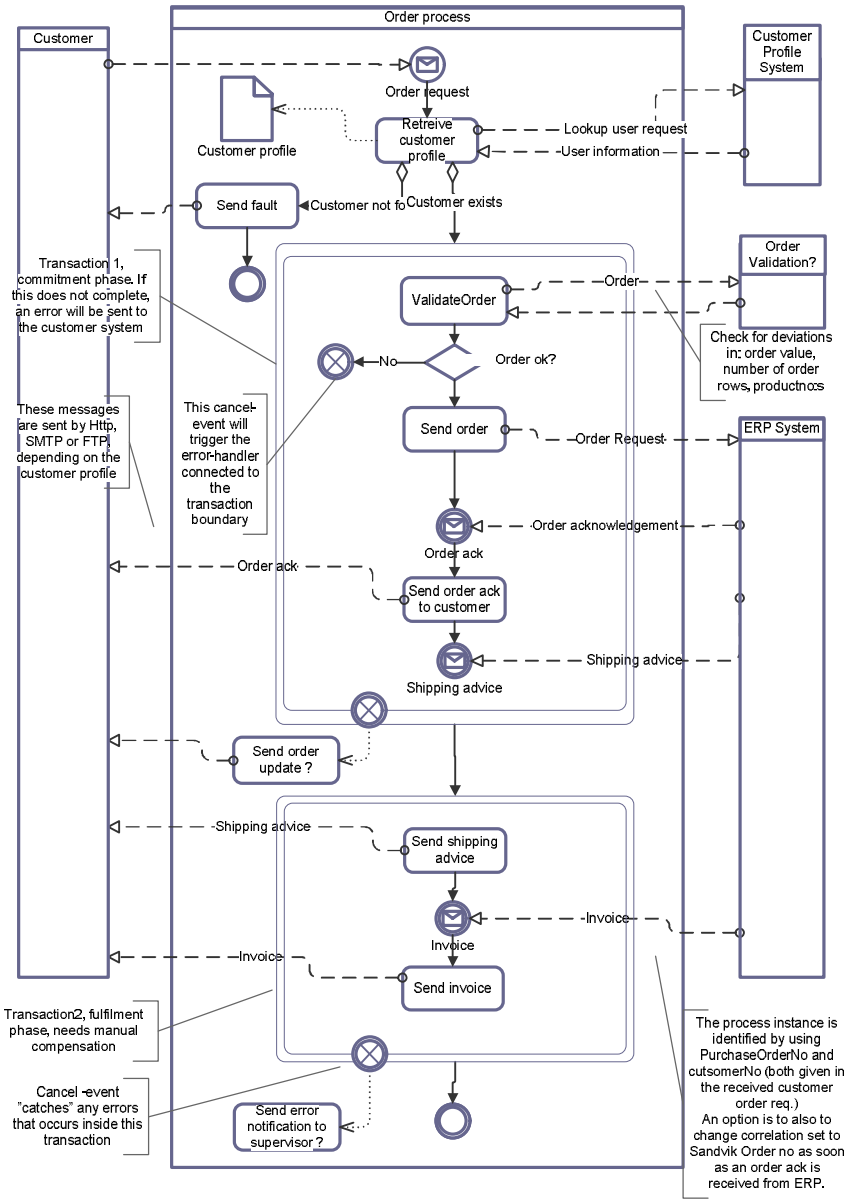

Figure 4, The order process with transactional and informational aspects

6 References

White, S. “Business Process Modeling Notation” Version 1.0, The Business Management Initiative, May 2004.

Henkel, M, Zdravkovic, J, “Service-based Processes – Design for Business and Technology”, Serviam Research Paper, 2004.

Jablonski, S. “A Software Architecture for Workflow Management Systems”, In Proceedings of the Ninth International Workshop on Database and Expert Systems Applications (DEXA’98) (Vienna, Austria, August 1998). IEEE Computer Society, 1998, 739-744.

Rausch-Scott, S. “TriGSflow – Workflow Management Based on Active Object-Oriented Database Systems and Extended Transaction Mechanisms”, PhD Thesis, University at Linz, 1997.

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