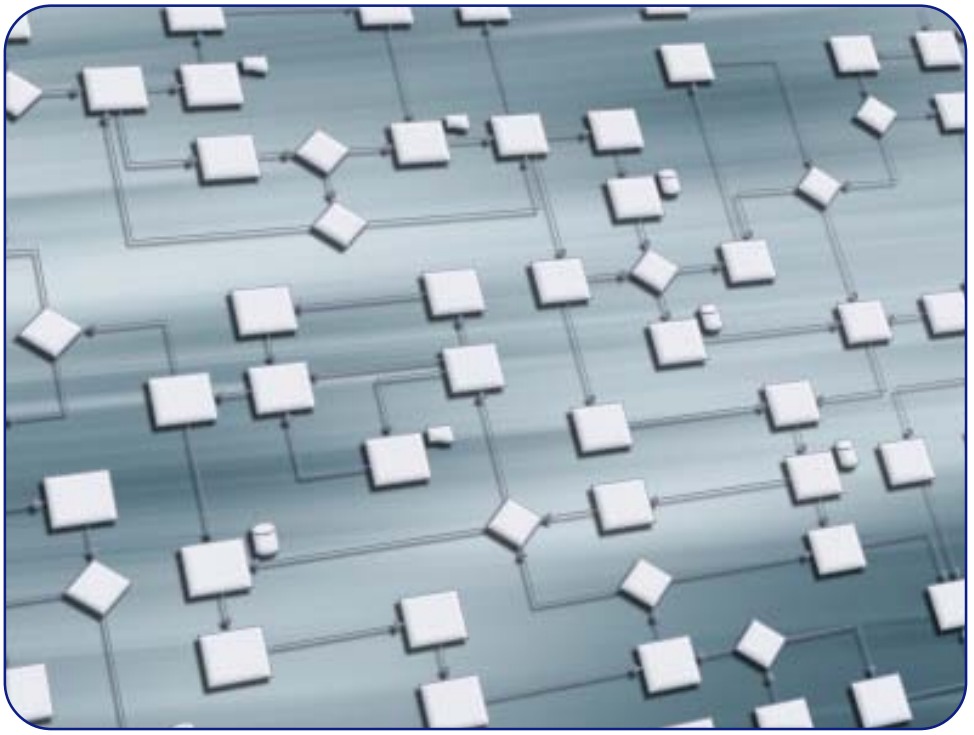
ESPOO 2007 VTT RESEARCH NOTES 2407



Mika Koskela & Jyrki Haajanen

Business Process Modeling and Execution

Tools and technologies report for SOAMeS project

VTT TIEDOTTEITA – RESEARCH NOTES 2407

Business Process Modeling and Execution

Tools and technologies report for SOAMeS project

Mika Koskela & Jyrki Haajanen

ISBN 978-951-38-6958-8 (URL: http://www.vtt.fi/publications/index.jsp)
ISSN 1455-0865 (URL: http://www.vtt.fi/publications/index.jsp)

JULKAISIJA – UTGIVARE – PUBLISHER

VTT, Vuorimiehentie 3, PL 1000, 02044 VTT
puh. vaihde 020 722 111, faksi 020 722 4374

VTT, Bergsmansvägen 3, PB 1000, 02044 VTT
tel. växel 020 722 111, fax 020 722 4374

VTT Technical Research Centre of Finland, Vuorimiehentie 3, P.O. Box 1000, FI-02044 VTT, Finland
phone internat. +358 20 722 111, fax +358 20 722 4374

VTT, Vuorimiehentie 3, PL 1000, 02044 VTT
puh. vaihde 020 722 111, faksi 020 722 7024

VTT, Bergsmansvägen 3, PB 1000, 02044 VTT
tel. växel 020 722 111, fax 020 722 7024

VTT Technical Research Centre of Finland, Vuorimiehentie 3, P.O. Box 1000, FI-02044 VTT, Finland
phone internat. +358 20 722 111, fax +358 20 722 7024

Cover image: Stockxpert

Technical editing Maini Manninen

Koskela, Mika & Haajanen, Jyrki. Business Process Modeling and Execution. Tools and technologies report for SOAMeS project. Espoo 2007. VTT Tiedotteita – Research Notes 2407. 63 p. + app. 2 p.

Keywords business process modeling, business process execution, business process management, service-oriented architecture

Abstract

This report presents the results of a survey on business process modeling and execution technologies. The first phase of the research consisted of a broad survey on the available language options. For business process execution, Business Process Execution Language (BPEL for short, officially WS-BPEL or BPEL4WS depending on the version) was considered as the only relevant option. Other executable languages were either obsolete or academic proposals not suitable for industry use. For business process modeling, Business Process Modeling Notation (BPMN) and UML Activity Diagram (AD) were considered suitable. Other available options did not provide enough support for transformations to executable languages.

The expressive power of the languages was evaluated by comparing how well the languages support different workflow patterns. It was found out that there is a significant gap between the expressive power of modeling and execution languages, which means that all models cannot be directly transformed to executable code. Between BPMN and UML AD, the differences in pattern support were minimal. However, it was noted that the specifications are partly ambiguous, which can lead to misinterpretations in the transformations.

The practical utility of the findings was demonstrated by testing two available tools that supported BPMN and BPEL and that were considered prominent based on their documented functionalities. The test results showed that the transformation functionalities were to a large extent dependent on the expressive power of the languages. It was concluded that the technologies have not yet fully matured, but first steps in their adoption can already be taken, because by taking the known shortcomings of the technologies into account in the modeling, automatic transformations from models to code, and even vice versa, can be realized.

Preface

This report describes the research conducted in SOAMeS (Service Oriented Architecture in Multichannel e-Services) in VTT Technical Research Centre of Finland in collaboration with University of Helsinki. The project started in mid 2006 and it will be finished during the year 2008. One objective of the project was to evaluate the feasibility of existing technology that is used in service oriented environment. These include, for example, business process modeling and execution languages. This report focuses on tools and technologies on this domain.

The target audience of this document includes enterprises that are adopting business process modeling and execution technologies and tools. Additionally, our research is relevant to developers who work with the focal technologies or build custom tools based on the discussed standards. This report should be valuable to people on many levels of the organization. The introduction, conclusion and summary sections are accessible to managers who need information to support their decision making on IT strategy. The conducted tests and evaluations as well as the more detailed descriptions of the technologies should be examined by the IT specialists of the enterprise. By reading the more detailed sections of this report, the specialists can evaluate what kind of value the discussed technologies can provide and what kind of requirements does their adoption pose.

Contents

List of symbols

AD	Activity Diagram
BPD	Business Process Diagram
BPDM	Business Process Definition Metamodel
BPEL	Business Process Execution Language
BPEL4WS	Business Process Execution Language for Web Services
BPM	Business Process Management
BPMI	Business Process Management Initiative
BPML	Business Process Modeling Language
BPMN	Business Process Modeling Notation
BPMS	Business Process Management Suite
BPSS	Business Process Specification Schema
EPC	Event Driven Process Chain
ICT	Information and Communication Technologies
EPML	Event-Driven Process Chain Markup Language
IDEF3	Integrated DEFinition Method 3
MDA	Model Driven Architecture
OMG	Object Management Group
OWL-S	OWL-Services
PIP	Partner Interface Process
PNML	The Petri Net Markup Language
RAD	Role Activity Diagram
SOA	Service-Oriented Architecture
UML	Unified Modeling Language
WS-BPEL	Web Services Business Process Execution Language
WS-CDL	Web Service Choreography Description Language
WSCI	Web Service Choreography Interface
WSCL	Web Service Choreography Language
WSDL	Web Service Description Language
WSFL	Web Services Flow Language
XMI	XML Metadata Interchange
XML	eXtensible Markup Language
XPDL	XML Process Definition Language
XSD	XML Schema Document
YAWL	Yet Another Workflow Language

1. Introduction

Modern enterprises are faced with global rivals and a tightening competitive environment. At the same time the changes in the business environment and daily routines are constant and occur at increasing pace. To survive in these conditions companies are focusing on their core competencies and outsource other functions to their business network partners for whom these functions are core competence. Hence, modern business life is becoming increasingly networked. This networking is driven and enabled by the recent rapid development in the Information and Communication Technologies (ICT). Especially the Internet based communications and business process and information system integration possibilities that became available since the mid 1990's have contributed to this development.

The plain focus on core competencies is not enough in competitive business conditions. Companies need to keep fit and develop their processes and infrastructure constantly. This also means that their customers and business partners will require constant changes on their common business processes and information system interoperability. At the same time, information systems and business architectures of the companies are going through changes. The keywords for these changes are: loose-coupling, reusability, isolation of technology and business requirements, interoperability and automation. Once these changes have taken place, we will have totally different looking enterprises with less or no stove-pipe departmental information systems, monolithic applications, etc. At this point, the time is ripe for the introduction of next generation enterprise computing, based on model driven business process management, execution and integration.

1.1 Background

Business Process Management (BPM) is an interdisciplinary domain which deals with the continuous development of business processes with the help of IT. BPM has gained lots of popularity in recent years due to the rise of the Service-Oriented Architecture (SOA) paradigm and related technologies that can be applied to execute and manage IT-enabled business processes. To align the business requirements with the executable processes, the modeling should start from high level processes, which would be refined to more detailed and exact models and finally to executable code. This development is outlined in Figure 1. To improve productivity of the IT organization and maintainability of the models, the transformations between models and code should be automated as much as possible. This kind of approach can be related to the Model Driven Architecture (MDA) framework, although the MDA-related technologies are not necessarily used. In this report, it is evaluated how well the state-of-the-art BPM technologies fulfill the promise that executable code could be automatically generated from business process models.

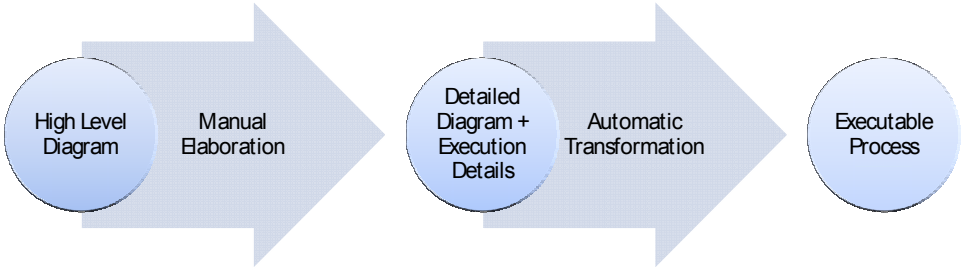


Figure 1. Development from business process diagram to executable process.

1.2 Motivation and objectives

As a part of the SOAMeS project, the state-of-the-art in business process modeling and execution is examined. Recognizing and managing business processes in companies is not a new phenomenon, but business process automation and integration have gained momentum in recent years, much because the enabling SOA-related standards and technologies. Mastering BPM is not a straightforward task, however: according to Gartner, only a small proportion of companies adopting BPM will succeed in stages beyond intra-process automation and control in the near future (Melenovsky & Sinur 2006). Enabling and supporting IT solutions are one of the success factors in this maturity model, but for many companies this field can be confusing due to the wide variety of available options and contradicting promises and claims made by different instances. It is, therefore, important to find out what can be expected from these technologies today, and to give directions for future BPM-related IT investments.

The purpose of this research is to examine which technologies are currently most prominent for business process modeling and execution in industry use. In the survey, a special emphasis is put on the relationship between modeling and execution languages. In other words, it is examined how well business process models can be transformed to executable processes and, to some extent, vice versa. This kind of research is considered relevant, because the requirements and possible limitations are not clearly stated by tool vendors or the standardization organizations. A complementary goal is to outline the relationship between BPM as a management domain and the topical technologies. For example, this means assessing the role of the state-of-the-art modeling standards in companies modeling and managing business processes: in order to really provide value to the organization, the notations that align with the executable processes should be accessible by both IT and business users.

The specific phases and objectives of the research include:

- to clarify the current state of the languages in the Business Process Management domain

- to evaluate the most prominent standards in business process modeling and execution languages based on their expressive power, suitability and tool support; a special emphasis is put on how well they could enable transformations between modeling and execution languages
- to identify potential problems in transformations based on gaps in expressive power; this is done by discussing the existing research on the *workflow pattern* support of the languages
- to demonstrate the transformations and their possible shortcomings using prominent tools
- to provide guidelines for companies adopting BPM tools now and in the near future.

It should be pointed out that the purpose of the expressive power analysis is not provide a complete evaluation of each language independently, but to point out the possible differences between the languages and to act as a background for the tool testing. An overview is still given, and a more detailed investigation can be carried out with little effort by business IT experts, using the introduced earlier research. Also, it should be noted that the purpose of the tool test is not to provide purchasing recommendations. The introduced tools are advanced and good candidates for experimenting with the technologies, but a valid tool comparison would have required more candidates and different criteria for tool selection and evaluation than was used in this research. The field is also constantly evolving, which means that in-depth tool benchmarking can only be considered valid for a relatively short period of time. However, based on our experiences, guidelines for tool selection are formulated. Additionally, the recognized shortcomings can be used as criteria for more specific and demarcated tool evaluation in companies.

2. Business process modeling and execution

2.1 Modeling languages

Visual business process modeling, i.e., drawing a business process diagram, is ideally the first step of the business process lifecycle. Numerous notations are available for this purpose and on the highest abstraction level, one can even visualize the core diagram elements using arbitrary shapes and lines. Also XML-based languages can sometimes be categorized as modeling languages. However, in this research we investigated visual business process modeling languages that are detailed enough to express executable processes so that code could be generated based on the diagrams. We use the term *modeling language* instead of *modeling notation* in order to stress that the discussed standards aim at specifying the detailed behavior that the modeling elements indicate, not only the visual notation.

2.1.1 Available options

To narrow down the options for modeling languages, we investigated languages that were discussed in (List & Korherr 2006). For a table comparing these languages, see Table 5 in Appendix A. From their list, we omitted proprietary languages, academic proposals, languages with no specified visual notation and languages that were not intended for modeling executable processes. This left us with two prominent options: Business Process Modeling Notation (BPMN) (Object Management Group 2006) and UML Activity Diagram (AD) (Object Management Group 2005). These languages were compared based on their expressive power, status and support. Expressive power refers to the language's ability to present different kinds of process constructs, patterns and situations that appear in business processes. It was evaluated based on existing workflow patterns research (Section 3). The later section provides more detailed discussion on the topic. At this point, we conclude that the differences in the expressive powers of BPMN and UML AD are not significant. In some situations, BPMN provides better expressive power than UML AD, however. It should also be noted that for both modeling languages, the specification is in some cases too vague and it is thus unclear whether certain patterns can be expressed.

The deciding factor for this study was thus tool support: very few tools for generating executable business process code from UML Activity Diagrams were available. The original organization responsible of BPMN also stated that UML has a different viewpoint on business processes, i.e., it focuses more on software design (OMG.org 2004). Although this statement must be taken with caution because it was given by the developers of another standard, we do think that BPMN is more easily accessible for both business and IT users. This can be considered an important factor in business process modeling.

2.1.2 Business Process Modeling Notation (BPMN)

BPMN is one of the youngest business process modeling languages available but it has already gained a notable amount of popularity nevertheless. At the time of writing, there are 43 BPMN implementations, i.e., tools which the standardization organization, the Object Management Group (OMG), considers to support BPMN, and many of them support BPEL generation (OMG.org 2007). The first BPMN specification was released in May 2004 by BPMI.org. The current version is 1.0 Final Draft, driven by Object Management Group which merged with BPMI.org (Object Management Group 2006).

From one perspective, the aim of BPMN is to provide a notation that both IT and business users understand. This means that the basic elements of the language should be easy to access. From a second perspective, BPMN is designed to act as a visual notation for executable languages. BPMN is able to present *private processes*, *public processes* and *collaboration processes*. The idea is simply that one can freely choose the level of granularity to use when modeling processes with BPMN. Most BPMN elements can be mapped to execution but some are used purely for informative purposes. One of the shortcomings of BPMN is that it lacks formal semantics, and the specifications for certain elements can also be considered inadequate for execution purposes. Additionally, the specification does not include an XML interchange format for BPMN diagrams. For this purpose, OMG has subsequently introduced the Business Process Definition Metamodel (BPDM) specification but it is not yet supported by the available tools.

2.1.2.1 BPMN basics

In this report, we introduce the basics of BPMN so that readers, who are not familiar with it, capture the general idea of BPMN modeling and the research we have conducted. The detailed discussions in the following sections often require more in-depth understanding of BPMN and the underlying execution language. We do not cover all the necessary details here. Instead, we target those sections to people with basic knowledge of the technologies. However, it can be said that learning BPMN does not require tremendous efforts, especially if one is familiar with some business process modeling language. In addition to the specification (Object Management Group 2006), we recommend the introductory article (White 2004a) and the BPMN tutorial (White 2006). The author of the both articles has also contributed in the official specification and these documents are available on the official BPMN web site[1].

[1] http://www.bpmn.org.

In BPMN, a Business Process Diagram (BPD) is a visual presentation of a business process. The term *model* is also often used to refer to the outcomes of BPMN modeling. The specification does not make a clear distinction between these concepts. However, the term model usually implies that also the necessary execution details (e.g., variable information) have been specified, whereas the concept of Business Process Diagram refers only to the graphical constructs. A BPD consists of elements whose visual appearance and semantics is defined in the BPMN specification. The elements consist of *Activities,* and the actors carrying out those activities. The actors are represented by *Pools* which can be further divided into sub-actors by *Lanes*. These are used on purely informative purposes and they do not affect the execution of the process. Additionally, the progress of the process is determined by *Events*. Inside a pool, which often represents an independent IT system, the sequence of activities is represented by *Sequence Flow*. Interaction with other pools is indicated by *Message Flows*. The progress of the *Sequence Flow* is often described using the concept of a token which passes through the flow objects. When discussing the control flow of the process, the term *upstream* is used to refer to the elements or tokens that appear or are generated before the discussed point in the process. Similarly, the term *downstream* refers to the parts that follow the discussed point. The control flow can diverge based on specified conditions. The conditions are directly attached to the flow or different kinds of *Gateways* are used to indicate decision points. These basic elements are depicted in Figure 2.

2.1.2.2 Semantics of the elements

BPMN defines several special versions of these elements. For example, *Activity* can be a sub-process or a looping activity. In turn, an *Event* can be a start event, intermediate event or an end event and it can represent an incoming or an outgoing message or an error. Besides activities and their connectors, BPMN defines *Artifacts* such as *Data objects* which can be annotated to indicate the inputs and outputs of the activities. (Object Management Group 2006)

The *End Event* indicates when the process will end. BPMN includes different kinds of *End Events*, which define different results for the process. A very typical end event is a *Message End Event*, which implies that a message is sent as the outcome of the process. Other types include, e.g., *Error* (an error is thrown), *Compensation* (the process is rolled back), *Terminate* (all activities in the process immediately end) and *Empty* (the result is not specified). *Terminate End Events* are special because when they are triggered, all activities are ended. In the case of other type of *End Events*, all tokens for the process have to be consumed by an *End Event* before the process instance is considered completed.

In BPMN, *Gateways* are used to control the sequence flow and in, more complex situations, the use of them is compulsory. There are two basic types of gateways: *Data-based* which evaluate the chosen path based on, e.g., values of variables and *Event-based* which depend on, e.g., an arriving message. They can be *splitting, forking merging* or *joining*, i.e., they can split one sequence flow to several (alternative or parallel) paths or combine them together. Forking or merging without a gateway is referred as uncontrolled. The gateways can be *inclusive* (OR), *exclusive* (XOR), *complex* or *parallel* (AND). In exclusive gateways, the sequence flow can take only one of the outgoing paths. In merging, they are often not required. Parallel processing creates multiple tokens which are combined back together at the merging phase. In inclusive gateways, all of the combinations of the independent paths can be taken. Parallel gateways define multiple paths that are executed concurrently. In forking, they can often be left out. In merging, they indicate that all of the inputs have to be available before the execution is continued. Complex gateways can be used to combine the behavior of several linked gateways. The forking or merging behavior of the complex gateway can be determined by an expression which can, e.g., evaluate process data to determine which flows to select. (Object Management Group 2006)

The behavior indicated by different kinds of forks and joins should be understood in order to understand the test case examples in Section 5. In Tables 1 and 2, the semantics of different splits and joins in BPMN are described informally. Example diagram constructs are presented in Figures 3 and 4. It should be pointed out that the join number one is only a special case of join number two, but it is presented separately in order to clarify the behavior indicated by unsynchronized merge in BPMN.

It should be noted that in many cases the exactly same behavior can be realized both by gateways and uncontrolled flow. An example of this is given in the context of AND-split Figure 3. In OR- XOR-splitting, C*onditional Sequence Flow* (arrows with attached diamonds) must be used if gateways are omitted. The attached conditions tell whether or not the choice is exclusive (see for example the XOR-split in Figure 3 – it is implicit that either yes or no can be taken, but not both). It is often more clearer to use gateways, because, for example, the gateways explicitly indicate that the alternative paths are exclusive. If uncontrolled flow is used, this restriction must come up in the conditions attached to the control flow. One should also note that the exclusive (XOR) gateway can be modeled with or without the drawing in it – the indicated process behavior is exactly the same.

Table 1. Process behavior indicated by BPMN splits according to (Object Management Group 2006).

Construct	**Explanation**
Split: XOR-split (exclusive OR)	The control flow splits into more than one branch. Only one of them can be chosen during runtime.
Split: OR-split (inclusive OR)	A branching point where all paths are independent: all combinations of the paths can be taken.
Fork: AND-split (parallel split)	A branching point where a single path of sequence flow is divided into two or more parallel paths. A token is immediately sent to each of the outgoing *Sequence Flow*.

Table 2. Process behavior indicated by BPMN splits according to (Object Management Group 2006).

Construct	**Explanation**
Merge: XOR-join	Combining multiple paths of sequence flow into one. The process shall continue when any of the incoming tokens arrives to the joining point, without considering other incoming paths.
Merge: XOR-join of parallel activities	The process will continue immediately when one of the incoming flows produces a token. There will be more than one incoming token, which means that an instance of the merging activity will be created for each of them.
Merge: OR-join	Combining multiple paths of sequence flow into one. The process will wait for all the tokens produced upstream to arrive before continuing.
Join: AND-join	Synchronizing multiple paths of sequence flow. The process will wait for all of the incoming paths to produce a token before continuing. In other words, all paths must be available.

Based on these tables, it can be said that the splitting behavior of BPMN is very clear and easy to understand. The joining behavior, however, is not so straightforward and it is likely to create misunderstandings among people who are familiar with other modeling notations. For example, it is not very obvious that sequence flows that merge without gateways proceed independently of each other, thus creating multiple instances of their common activity if the flows are parallel. In UML AD, for instance, a merge node is used in these kinds of situations. Additionally, based on the specification document, it is not exactly clear how the number of possibly incoming tokens is

determined. The OR-join has been stated to be inadequately specified, e.g., by Ouyang et al. (2006). Because of this, it was also not used in the test case processes. However, based on these kinds of issues, it must be noted that the standard has not fully matured yet. Also, because formal semantics is not included in the specification, it is possible that tool vendors misinterpret the rules. This results in potentially incorrect enactment behavior, although executable code could be created based on the model.

It should also be pointed out that if the use of splits and joins is not carefully analyzed, erroneous models can be created. Based on these models, accurate executable code may even be generated, but the problems occur when the processes are being executed. Three typical classes of problems can be identified: deadlocks, livelocks and multiple instances. In a deadlock, the execution ceases because the process waits for tokens which will never be generated. This can happen, e.g., if OR- and AND-splits are wrongly paired. In a livelock the process constantly changes its state, but does not progress, e.g., loops infinitely. Multiple instances of the same activity can lead to unnecessary resource consumption, or even incorrect business results.

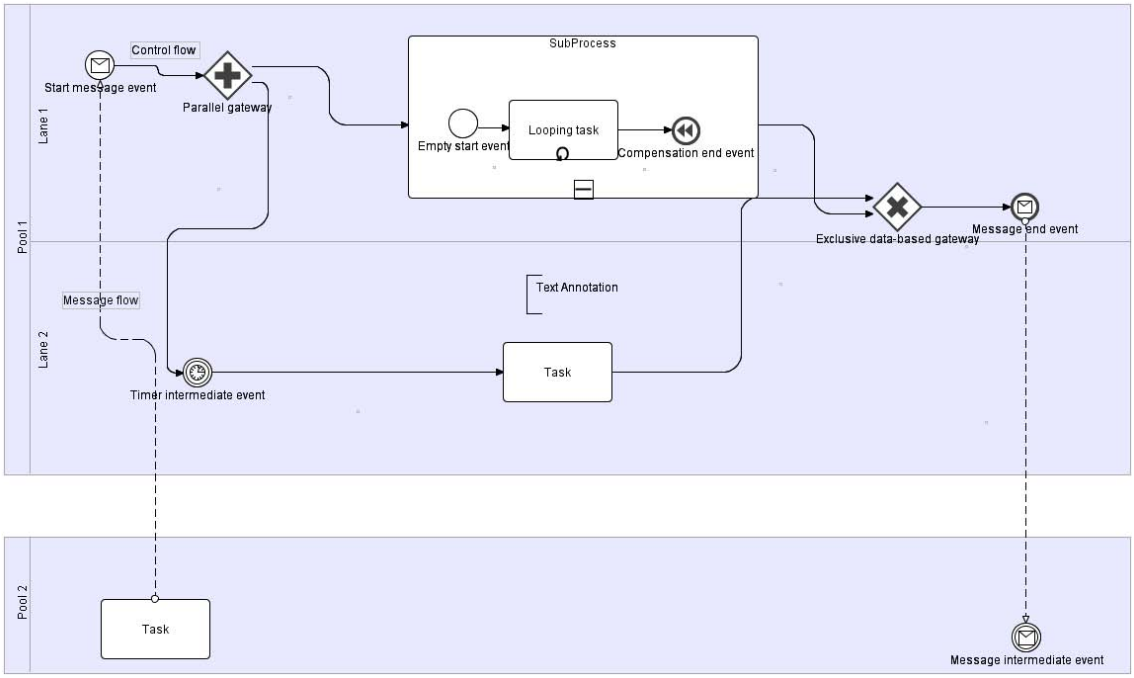


Figure 2. Examples of BPMN elements.

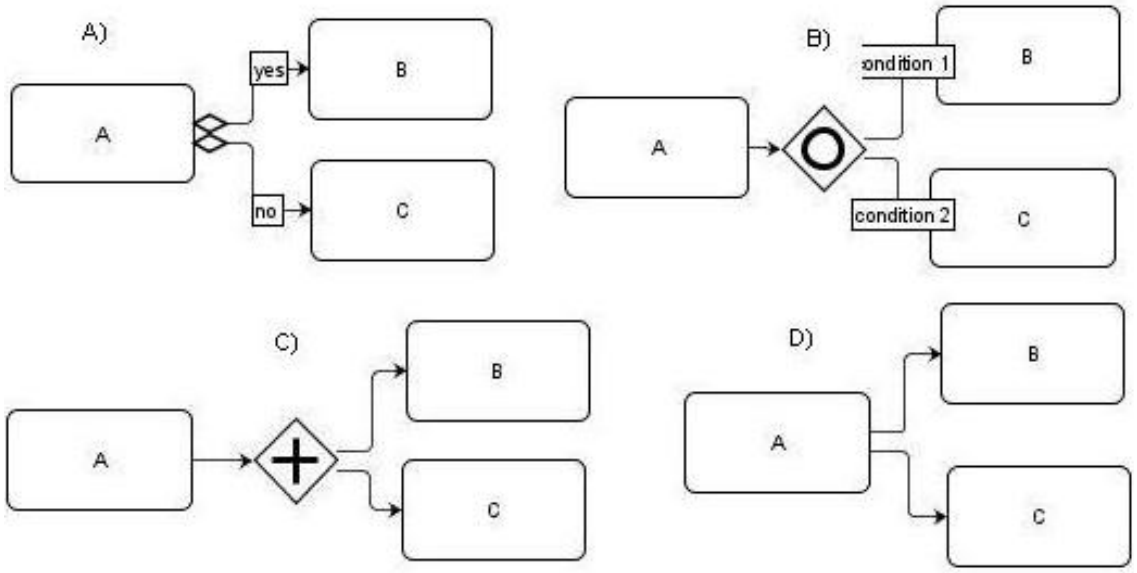


Figure 3. BPMN splits: A) XOR B) OR C) & D) AND (equivalent behaviour).

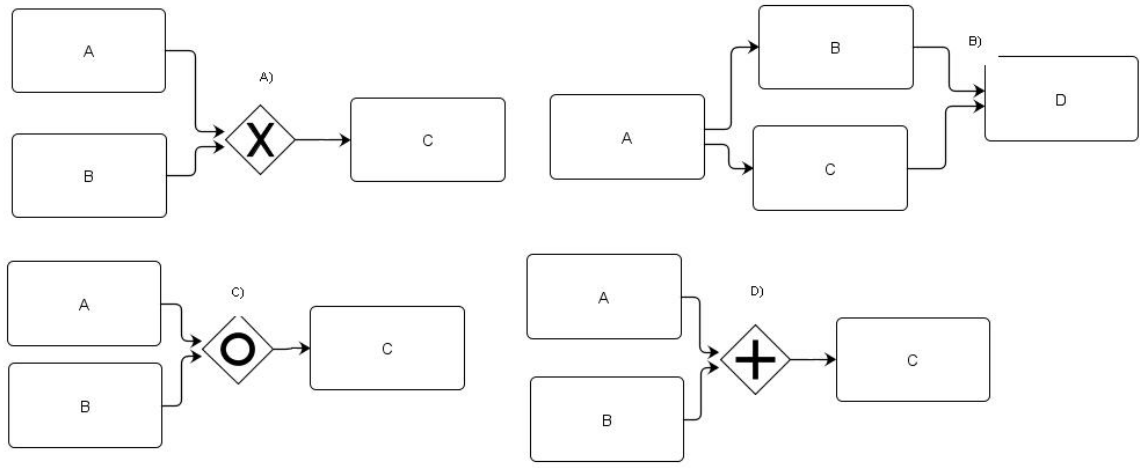


Figure 4. BPMN joins: A) XOR B) XOR (parallel activities) C) OR D) AND.

2.2 Execution languages

In the context of the SOAMeS research project, business process modeling focuses on processes that will be automated and executed in Service-Oriented environment. Therefore, the study on business process modeling tools also involves investigating executable business process languages.

2.2.1 Available options

The business process management domain includes several XML-based languages, but few of them can be actually used to execute automated business processes without translating them into another format. In this research, we first briefly inspected the languages listed in a relatively comprehensive survey by Mendling et al. (2004). From this list we omitted choreography languages, pure academic proposals, visual modeling languages and clearly superseded options. We then studied Business Process Execution Language (BPEL), Business Process Modeling Language (BPML), Business Process Definition Metamodel (BPDM) and XML Process Definition Language (XPDL). The only language that was considered suitable for automated execution was BPEL. Other three languages were obsolete (BPML) or focused more on process interchange than automated execution (BPDM and XPDL). XLANG and WSFL are also both options for execution languages, but they are directly included in BPEL versions that are currently in use. To limit the scope of the stidy, academic proposals such as Yet Another Workflow Language (YAWL) were excluded from this research. Therefore, BPEL was considered the only relevant execution language for this research. The categorization of the XML-based languages is summarized in Table 7 of Appendix A.

2.2.2 Business Process Execution Language (BPEL)

Business Process Execution Language (BPEL4WS 1.1 and WS-BPEL 2.0) (Andrews et al. 2003; OASIS 2007) was introduced in previous SOAMeS research notes (Kanniainen & Haajanen 2007). In short, BPEL is an XML-based language for specifying business processes in Web Service environment. It is used together with Web Service Description Language (WSDL) and other related technologies. This means that BPEL is used to define how the business process is built from Web Service invocations and what kind of interaction with external participants does the process involve.

BPEL can be seen as an XML-programming language for Web Service compositions. Like most programming languages, BPEL is complex and the specification is extensive. In-depth understanding of BPEL requires software development competence as well as knowledge of the Web Service technologies it is built on. Therefore, we only provide a brief introduction to BPEL here, in order to provide a basis for latter sections. For those who want to master BPEL, several books have been published. For an online introduction, see, e.g., (Miguel 2006).

The current BPEL standard is the result of combining different specifications developed by different organizations. The BPEL4WS language was originally proposed by Microsoft and IBM in July 2002. BPEL4WS combined properties from Microsoft's

WSFL and IBM's XLANG. The revised version of BPEL4WS (Andrews et al. 2003) has been widely adopted by tool vendors. In version 2.0, the language was renamed to WS-BPEL which was approved as an OASIS standard in 2007 (OASIS 2007). In this report, the acronym BPEL is generally used to refer to both versions because the older version cannot be considered obsolete. Clear distinction between the versions is made whenever necessary. The new version involves syntactic changes and improved alignment with other XML technologies such as XPath. Therefore, BPEL processes are not directly backward compatible.

2.2.2.1 Principles of BPEL

BPEL can be used to specify both executable and abstract processes. In the first case, the full implementation logic of the process is defined, whereas in the latter case only the message exchange between process participants is included. The core elements of a BPEL document include:

- *Roles* of the process participants
- *Port types* required from the participants
- *Orchestration:* the actual process flow
- *Correlation information:* definition of how messages can be routed to correct composition instances, i.e., how to map abstract specifications to running instances.

In this report, we focus on executable aspects of BPEL, for three reasons. Firstly, mappings between the modeling languages and BPEL do not currently cover abstract processes. Secondly, it has been convincingly stated that BPEL does not suit well for defining abstract processes (van der Aalst et al. 2005). This is because BPEL processes are always presented from the point of view of one partner. Consequently, business partners do not need to agree on using BPEL. Finally, the focus of this report is on executable processes, not incomplete process descriptions which are closely related to choreographies. Hence, abstract BPEL is also discussed in the choreography language report, which is also published within the SOAMeS research project. However, we describe the principles of abstract BPEL and, additionally, demonstrate the use of executable processes in collaborative settings in one of our test cases.

When transformations between business process languages are carried out, the adopted representation paradigm is an especially important question. Two basic categories can be recognized: block-oriented languages and graph-oriented languages. The first ones define the control flow by nesting different kinds of control primitives. The second ones specify it using different kinds of nodes and arcs. Arcs connect nodes with one another,

presenting temporal or logical connections. Transformations between these fundamentally different languages are problematic, because graph-oriented languages can express process patterns that block-oriented cannot. BPEL is for the most part block-oriented. Graph-oriented structures can be partially expressed using links. Modeling languages, in contrast, are often graph-oriented. (Mendling et al. 2006)

2.2.2.2 Basic BPEL elements

BPEL activities can be either basic or structured activities. The basic activities correspond to actual components of the process. They are realized through Web Service interaction, i.e., through invocations of WSDL operations. These activities include for example the following operations:

- <invoke>: calls an operation
- <receive>: waits for an input message
- <reply>: sends an output message
- <assign>: updates variable values
- <wait>: blocks the execution for a certain period of time.

The structured activities resemble control structures of conventional programming language. They constitute the block-oriented part of BPEL, which originates from XLANG. Structural activities of BPEL include:

- <sequence>: set of activities to be executed in the listed order
- <switch>: condition-activity pairs from which the first activity with true
- condition is executed. In WS-BPEL 2.0, this activity is replaced by <if> construct (OASIS 2007, p. 263)
- <pick>: list of event-activity pairs. When the first event from list occurs, the corresponding activity is executed. Handling of race conditions is not specified.
- <while>: one activity and condition. The activity is executed while the condition is true
- <flow>: the activities inside this activity are executed in parallel. A flow is completed when all of the parallel activities are completed.

Additionally, BPEL specifies handlers for events and faults. For each handler, an event, a scope and a corresponding activity to handle the event is defined.

The order of execution inside a <flow> element can be controlled using <link> elements. This defines the graph-oriented nature of BPEL, which originates from

WSFL. Consequently, BPEL links have an important role when business process diagrams are transformed into executable processes. In using BPEL links, the following restrictions apply, however (OASIS 2007):

- links cannot cross the boundaries of repeatable constructs such as <while>, and in WS-BPEL 2.0 only outbound links can cross <catch>, <catchAll> and <terminationHandler> scopes
- a <link> declared in <flow> cannot create a control cycle.

2.2.2.3 Abstract BPEL

In business-to-business settings, one is often interested in capturing the interactions between participants and messaging sequences without revealing the internal processing details for each participant. Abstract BPEL captures these kinds of processes. Essentially, abstract BPEL processes differ from executable processes by allowing internal details to be omitted from the process description. Thus, the process cannot be directly executed, but one can build different kinds of executable processes that correspond to the defined abstract process. In abstract BPEL, process information can be left unspecified in two ways: *omission* or *opacity*. In short, omission is an implicit form of opacity. This means that, for example, the variable reference attributes for web service invocations can either be left out or explicitly defined as *opaque*.

WS-BPEL 2.0 (OASIS 2007) specifies the role of abstract processes in more detail than BPEL4WS 1.1 (Andrews et al. 2003). According to (OASIS 2007, p. 147), abstract processes have different kinds of use cases. Therefore, the specification defines a Common Base for abstract processes. It can be refined by defining *profiles* for specific use cases. In the Common Base, all activities, expressions, attributes and data sources can be opaque. The profiles can restrict these opacity rules to make the process descriptions accurate enough for the purposes of the use case. The WS-BPEL 2.0 specification defines two abstract process profiles: a profile for observable behavior and a profile for process templates. In short, the first profile is meant for controlling business process contracts and the second profile for a high-level process representation in an organization.

2.2.2.4 BPEL extensions

Several BPEL extensions have been introduced, for example BPEL4People for presenting human activities in BPEL processes and BPEL Subprocesses for reusable subprocesses. The status of these specifications varies and detailed study of them was

not included in this research. It should be pointed out, however, that relying on extensions decreases the portability of BPEL processes, meaning that a process created for one engine cannot necessarily be executed in another engine.

2.2.2.5 Conclusion on BPEL

In conclusion, we state that BPEL is a relatively powerful language with a strong position and it is currently the best possible option for executing business processes. This is mainly because it does not have any serious competitors in the industry, considering process execution. In comparison with conventional programming languages such as Java, BPEL provides powerful mechanisms for typical business process interactions such as long-term transactions, asynchronous messaging and parallel activities. This means that it would require much more effort and lines of code to express the process in a conventional programming language. The downside is the syntactic restrictiveness of BPEL which is one source of problems in the BPMN transformations.

Although BPEL is an important standard, it lacks some features and complementary languages are needed. For process execution in electronic business networks, better support for collaboration processes is required. Choreography languages such as WS-CDL should be used for this purpose and use BPEL to implement the internal processes. The term choreography refers to the collaborations between interacting parties, i.e., the global process that each participant needs to agree on (van der Aalst et al. 2005, p. 10). For example, based on the global definitions described in a WS-CDL document, each party can implement solutions that conform to their role in the global process (Kavantzas et al. 2004). Choreography languages are also investigated in SOAMeS project. The results are delivered in a separate report.

3. Workflow patterns and business process languages

Workflow patterns are a popular and systematic method for evaluating workflow languages and tools. The terminology can be confusing, because in this report we have discussed the modeling and execution of business processes. In the academia, different views exist on the definitions of a *workflow* and a *business process.* The distinction is partly based on the different formalisms adopted in these two camps. For example, it has been stated that, in contrast to workflows, business processes can represent more dynamic situations in which processes and the relationships among its participants evolve as the process executes (Smith & Fingar 2003). On the other hand, Business Process Management can be merely seen as an extension of Workflow Management, covering also other lifecycle phases than modeling and execution, such as process monitoring. For the purposes of this research, however, it is enough to point out that workflow patterns are fully applicable to the tools and technologies under discussion.

Informally, patterns describe workflow constructs that represent certain process setting, originating from business requirements. They resemble design patterns in software engineering. Originally, van der Aalst et al. (2003, p. 6) recognized four different perspectives for workflow patterns:

- *control-flow perspective*: activities and their execution order
- *data perspective:* process and business data on top of the control perspective
- *resource perspective:* human and device roles responsible for executing the activities
- *operational perspective:* elementary actions of activities.

Workflow patterns constitute an active area of research. Information on the patterns and their support is maintained by the Workflow Patterns Initiative[2] on their website. The number of different patterns in each category has increased since their first introduction and also new perspectives have been introduced: *exception handling perspective* deals with causes and control of exceptions in workflows and *service interaction patterns* describe the possible interactions between different processes or processes and resources.

Comparing languages based on workflow patterns can be used to evaluate their suitability and expressive power (van der Aalst et al. 2003, p. 7). A language may be able to express a pattern, i.e., have expressive power, but the resulting expression might be, e.g., very hard to understand, i.e., unsuitable. Different workflow patterns represent different settings. Therefore, all patterns are not equally relevant to all purposes and domains.

[2]http://www.workflowpatterns.com/.

According to van der Aalst et al. (2003, p. 6) the control flow and data perspectives include the most important patterns while the perspectives on resources and operations have a supplementary role. Based on its relatively short history, same can be stated about the exception handling perspective. Service interaction perspective can be considered relevant to SOA but its focus is more on choreography than on execution. Additionally, modeling issues have not been considered. Here we present control flow patterns because they constitute the dominant workflow perspective. Also the data pattern support of the languages is briefly discussed. These patterns act as a basis for the test cases in which both suitability and expressive power is evaluated in the context of language transformations.

The Workflow Patterns initiative has also evaluated the pattern support of certain workflow products, including, for example, WebSphere and SAP. Because none of these products is fully based on standard modeling and execution languages, the pattern evaluations are not further discussed here.

3.1 Control flow patterns

The original body of work (van der Aalst et al. 2003) defines 20 control flow divided in five categories. The work on control flow patterns continues and at the time of writing there are currently 43 pattern descriptions available on the Workflow Patterns initiative website. The new patterns are mostly based on the original ones so that an old pattern has been divided into several new patterns because of the different situations it may occur in. The original patterns have also been re-evaluated and revised to avoid ambiguities. The control flow patterns include: (Workflow Patterns Initiative 2007)

- 5 *basic control flow patterns:* elementary aspects of process control similar to the initial definitions of WfMC (e.g., sequence, parallel split)
- 12 *advanced branching and synchronization patterns:* more complex branching and merging (e.g., multi-merge)
- 5 *structural patterns:* behavioral design restrictions (cycles, termination)
- 7 *multiple instance patterns:* situations where multiple execution threads relating to same activity are active at the same time
- 7 *state based patterns:* situations dependent on the process state (defined by broad set of data)
- 5 *cancellation patterns:* variants of the activity cancellation concept
- 2 *triggers* to trigger activities by a signal from another part of the process or from the external environment.

The most important control flow patterns whose supports differ in the focal languages are summarized in Table 3. These act as a basis for recognizing possible shortcomings in the transformations between modeling and execution languages.

Table 3. Selected control flow patterns and their support according to (van der Aalst et al. 2003; Workflow Patterns initiative 2007).

Pattern name	**Description**	**BPEL 1.1**	**BPMN 1.0**	**AD 2.0**
Discriminator	The convergence of two or more branches into a single subsequent branch. Only one token is accepted, others will be blocked.	no support	partial; specification on join condition unclear (structured)	partial; specification on join condition unclear (structured)
Multi-merge	The merging of multiple parallel paths without synchronization.	no support	support	support
Arbitrary cycles	Cycles with more than one entry or exit point.	no support	support	support
Multiple Instances with a Priori Design-Time Knowledge	Creating a multiple instance activity when the required number of instances is known at design time.	no support; no support for multiple activity instances	support	support
Multiple Instances with a Priori Run-Time Knowledge	Creating a multiple instance activity when the required number of instances may depend on runtime factors but is known beforehand. instance creation	no support; no support for multiple activity instances	support	support
Explicit termination	Process is terminated if it reaches a certain state.	no support	support	support
Acyclic Synchronizing Merge	Two or more branches converge into a single subsequent branch. Synchronization is made based on the information locally available to the merge construct.	support	no support	partial; specification on join condition unclear
Structured Synchronizing merge	Same as above, but in a structured context, i.e., so that there is a join of the same type for each split.	support	support	no support; the specification on join condition to achieve this unclear

Based on the pattern comparison it can be concluded that graph-oriented visual modeling languages have better support of control flow patterns than the highly block-oriented execution language. Some contrary exceptions can be recognized, however. These include *Acyclic Synchronizing Merge*: in BPMN, for example, the OR-join is

used to realize synchronizing merge. However, it can only be used in *Structured Synchronizing Merge,* meaning that it assumes that each split in the model has a corresponding join of the same type. It should also be noted that in the most complex cases, the specifications are not detailed enough to guarantee full support for the pattern.

In addition to recognizing the gaps between modeling and execution, the patterns were examined in order to analyze the expressive power of the languages. This was based on Workflow Patterns Initiative (2007). For detailed listings of the pattern support, the reader is advised to study this web site. It can be concluded that BPEL supports relatively well the different control flow patterns, although some shortcomings can be identified. The relevance of BPEL's evaluation is hindered by the fact that it has no real competitors as an execution language standard in the industry. Clearly, BPEL's pattern support is the union of XLANG's and WSFL's supports, because the constructs of both languages are included (van der Aalst et al. 2005). BPEL4WS 1.1 is challenged by the proprietary workflow languages and the extended versions of BPEL, some of which have also been evaluated in (Workflow Patterns Initiative 2007). The differences are relatively small, however. For example, standard BPEL and IBM WebSphere Integration Developer support the same control flow patterns. Oracle BPEL introduces a new construct for multiple parallel instances of the same activity, which enables it support some multiple instance patterns that the standard does not support. On the other hand, e.g., a pattern called *Interleaved Parallel Routing* (parallel tasks with partial ordering requirements) cannot be implemented in Oracle BPEL although it is supported through <scope> in standard BPEL.

The control flow supports of the focal modeling languages, BPMN and UML AD, are almost equal. Also in this case, the *Synchronizing Merge* patterns are the ones to point out differences in expressive power. BPMN supports the structured version, as described above, but UML AD does not.

3.2 Data patterns

Data patterns capture the ways of utilizing and representing data in workflows. Together with control flow patterns, they constitute a basis for pattern-based evaluation of workflow (and business process) standards and tools. Data patterns can be divided into four categories (Russel et al. 2005, p. 3):

- *Data visibility:* how data elements can be viewed by different workflow components – 8 patterns described in (Workflow Patterns Initiative 2007)

- *Data interaction:* how data is communicated between the active elements in the workflow – 6 internal and 11 external interaction patterns described in (Workflow Patterns Initiative 2007)
- *Data transfer:* how data elements are transferred between workflow components and what kind of mechanisms exist for passing data elements across the interfaces of workflow components – 7 patterns described in (Workflow Patterns Initiative 2007)
- *Data-based routing:* how data elements can influence other aspects in the workflow, particularly the control flow perspective – 7 patterns described in (Workflow Patterns Initiative 2007).

The Workflow Patterns Initiative (2007) often defines many variations of the same data pattern. For example, there are both *Push-oriented* and *Pull-oriented* versions of all data transfer patterns. In general, both execution and modeling languages lack support for some versions of the patterns. The relevance of some of the patterns can thus be questioned to some extent.

BPEL provides support for all of the common data patterns. Problems are posed by, e.g., complex data transfer patterns. The proprietary alternatives do not provide better support, however. Also both modeling languages support all basic patterns, but shortcomings can be recognized in many advanced variations. BPMN provides a slightly better support than UML AD. This is mainly because in BPMN, *Pools* can be used to represent external environments in data transfer patterns – UML AD does not have such constructs. In case of data patterns, it cannot be so clearly stated that either modeling or execution languages would result in better expressive power. Some differences can be pointed out, however. For instance, modeling languages can both express preconditions for activities, but they cannot indicate data transfer by reference, like BPEL.

4. Existing research on BPMN-BPEL transformations

BPMN has had a close relationship with BPEL from the beginning and preliminary mappings to BPEL are provided already in the BPMN specification. Based on this, lots of research has been conducted considering the subject (White 2005; Recker & Mendling 2006; Ouyang et al. 2006; Emig et al. 2006) and several BPMN tools generate BPEL based on the diagrams. The basic mappings between BPMN and BPEL elements are also covered in the BPMN specification (Object Management Group 2006). We started our study on the subject by investigating the existing research. We found out that certain process patterns were constantly considered problematic in publications. These shortcomings in the transformations stem from the fundamental differences between graph-oriented and block-oriented languages. Strategies for overcoming these gaps exist, with their own limitations. Due to the vast amount of existing research and implementations, we did not try to develop a new algorithm for BPMN-BPEL transformation but to evaluate the limitations of the existing tools and to outline strategies for overcoming those. In this section we briefly introduce two of the most prominent transformation approaches found. The first one, (Ouyang et al. 2006), is a technical report by a group of well-known workflow scientists. Some of the advanced constructs of BPMN, such as exception handling mechanism, are not included in the algorithm, though. The second one, (Gao 2006), is an industry paper by the vendor of a prominent BPMN tool, eClarus.

4.1 Links and event handlers

Event handler is a BPEL construct which is familiar from many conventional programming languages. For each handler, an event, a scope and a corresponding activity to handle the event is defined. Any valid BPMN model can be translated to BPEL using event handlers. This means that for each BPMN task, event or gateway, a set of event handlers is defined. This way, any kind of parallel behavior can be managed despite the block-oriented nature of BPEL. The diagram still has to be free of, e.g., deadlocks and livelocks in order to produce a functional process. This approach, however, results in BPEL code which is not readable and hence difficult to modify by developers afterwards. However, two classes of BPMN models can be translated to BPEL models by using constrained control flow constructs of BPEL: structured and synchronizing process models. The models in the first category can be translated into the five structured control flow constructs of BPEL (sequence, flow, switch, pick and while). The models of the second type can be translated to BPEL using a control link structure. The BPEL code generated this way is much more readable than the one based on event handlers. In short, the translation method aims at maximizing the readability of the code by resorting to BPEL event handlers only if no other options exist. (Ouyang et al. 2006)

The authors of (Ouyang et al. 2006) point out that defining reverse transformations for BPMN diagrams involving the entire BPMN diagram would be a challenging task. Their method does not therefore enable round-tripping, even though BPEL-to-BPMN transformations can be considered much more straightforward than the opposite operation. In the software development process, BPEL code can be generated from business process models using the method described earlier. However, it is possible that the BPEL code is modified later on. At this point, methods for automatic BPMN modification would be feasible to keep the models consistent. Therefore, the challenge of reversibility is mentioned as a future research subject in the article. (Ouyang et al. 2006, p. 18)

The translation algorithm of Ouyang et al. (2006) has also been successfully implemented. The result is a piece of software named simply as BPMN2BPEL[3]. It is a command line tool, which takes an XML serialization of a BPMN model as an input and produces a BPEL document using the approach described in the article. The resulting BPEL descriptions have also been validated using Oracle BPEL Process Manager (Ouyang et al. 2006, p. 35).

4.2 Links and diagram restructuring

Gao (2006) states that the developers of eClarus have solved the problem of round-trip engineering between BPMN and BPEL. The details of this translation algorithm are not public but the principles of the method are described. It is also admitted in the paper that not all BPMN diagrams can be mapped to BPEL in an isomorphic way. According to Gao, however, it is possible to rewrite the diagram so that it is BPEL isomorphic, i.e., it can be precisely expressed using BPEL. This rewriting is said to be a difficult process of semantic analysis and the details of executing it automatically are not discussed. The transformation of BPEL isomorphic diagrams is based on the basic mappings and so-called static token flow analysis. In this approach, a flow token is assigned for each sequence flow and downstream tokens are inherited by upstream flows. Tokens are divided into sub-tokens and merged back according to the flow. By analyzing the signatures of the tokens, transformable patterns in the diagram can be recognized. For example, if a flow object has two incoming sequence flows, and the token of the first flow is a sub-token of the second flow's token, a loop structure can be recognized. This method can also be used to find non-isomorphic graph structures which necessitate rewriting the diagram.

[3] http://www.bpm.fit.qut.edu.au/projects/babel/tools.

5. Tool-based experiments on transformations

It can be said that tools have a significant role in business process modeling and execution. BPEL is a complex language and writing the code manually is a time consuming and error prone task. Because of the specialized nature of BPEL, it can be assumed that even the majority of software developers are unfamiliar with its details. This learning curve can be made gentler by appropriate use of tools. In general, business process modeling and execution tools can be divided into four categories:

- *Business process execution platforms:* tools that guide the design of BPEL processes and execute the process in a BPEL engine, which is technically an application server with BPEL execution capability.
- *Business process modeling tools:* tools that can be used to model processes in a standard or proprietary notation. No support for generating BPEL from the diagrams is provided and the modeling is not restricted by the capabilities of the execution language.
- *Business process modeling tools with transformation support:* tools which are able to export diagrams to BPEL code or import BPEL files to generate BPMN models. If support for both functionalities exists, the tool is close to supporting *BPMN-BPEL round-trip engineering*. These tools generate BPEL code but it has to be deployed and executed using a separate product.
- *Business Process Management Systems/Suites (BPMS):* these tools support the whole process lifecycle from modeling to execution, with possible simulation capability. They generate BPEL code from BPMN models. The executable process can be deployed to an execution engine which is integrated with the modeling environment. The execution engine may provide the capability to monitor a process execution using the corresponding diagram as a basis for visualization.

The first category of tools is introduced in (Kanniainen & Haajanen 2007). In this project, the focus is on automated business processes, which means that business process models need to have a clear relationship with executable processes. Therefore, the second category of tools is omitted here. From the selected tools, one (eClarus) falls into the third category and the other (Intalio BPMS) to the fourth category. The properties of the categories are summarized in Table 4. The table indicates whether the typical tools in these categories include certain functionalities. If a column is marked as *possibly*, this means that tools in the category may have this kind of support, but this is not necessary.

Table 4. Modeling and execution tool categories.

Functionality	Execution platforms	Modeling tools	Modeling/ transformation tools	Business Process Management Systems/Suites (BPMS)
Modeling language support	no	yes	yes	yes
Process execution support	yes	no	no	yes
Modeling-to-execution transformation	no	no	yes	yes
Execution-to- modeling transformation	no	no	possibly	possibly
Process simulation	no	no	no	possibly
Process execution monitoring	possibly	no	no	yes

5.1 Principles for tool selection

The selection of investigated tools was to a large extent based on lists maintained by the Object Management Group**.** Also industry articles (e.g., Gao 2006) and free search on the web were exploited. The used tools were selected based on available documentation and, in certain cases, preliminary use experience. In the selection, the following properties of the tools were emphasized:

- *Language support:* the support for BPMN and BPEL was required. For BPEL, both BPEL4WS 1.1 and WS-BPEL 2.0 supports were considered suitable.
- *Claimed transformation support:* the products were evaluated based on the transformation support stated in the documentation. All selected tools support some level of code generation based on process diagrams.
- *Availability:* the research was restricted by the fact that the products needed to be available for research purposes at low costs. All included products were available freely, as a trial version or under academic license.

5.2 Selected tools

Based on the previously described criteria, several suitable tools could be found. We did not include a wide range of products in our test cases, because this could have affected the quality of the tests due to the limited time scope. Additionally, as the previous sections show, the transformation functionality of the tools is largely dependent on the expressive power of the languages, and the results can thus be generalized more than

with proprietary BPM tools. Finally, the field is under constant change and new products and improved versions of the old ones are being released at a rapid pace. Hence, the results of extensive comparisons would not have been feasible for very long.

We did not choose the tools randomly, however. The first tool, Intalio BPMS, was selected because it was freely available and because it provided the whole BPMS functionality. We feel that freely available products have currently an important role, because the technologies and the domain in general are at a relatively young stage. Organizations may need to assess whether such technology would provide considerable value, before making significant investments in tools. This kind of analysis is also suggested by Gartner in their analysis on Intalio (Hill & Drakos 2006). Accordingly, Intalio is the first credible BPMS product, although commercial products include a wider set of features that are accessible also to business users.

The second tool, eClarus Business Process Modeler for SOA Architects, was selected because it was one of the few tools that supported both BPEL export and import. The industrial paper covering the principles of its transformation method (Gao 2006) also increased our interest on this product. It is a commercial tool but the vendor provided us with a limited license for our research purposes.

5.2.1 Intalio BPMS Community Edition

Intalio BPMS Community Edition[4] enables experimenting with a full-blown BPMS without costs and therefore, it was chosen for this study. The functionalities are divided into two separate products: Intalio Designer for modeling business processes and Intalio Business Process Management Server for executing and managing them. The Community Edition of Intalio can only be used with a MySQL relational database and Apache Geronimo application server. The Enterprise Edition[5] does not have these restrictions. For the purpose of this research, i.e., mainly for examining the BPMN transformations, the Community Edition is considered completely adequate.

In this research, version 4.4 of both Intalio Designer and Intalio Business Process Management Server were used. In the early stage of the empirical study, version 5 of the Designer also reached alpha stage. Due to its immature stage, it could not be included in the research. According to the documentation on the web site, the tested version of the Designer requires Windows XP or Windows 2000. The Server will also work on Linux or Mac OS X. The toolset was only tested in Windows XP environment.

[4] http://bpms.intalio.com.
[5] http://www.intalio.com.

The basic modeling view of Designer is presented in Figure 5 and the listing of deployed processes in Server in Figure 6.

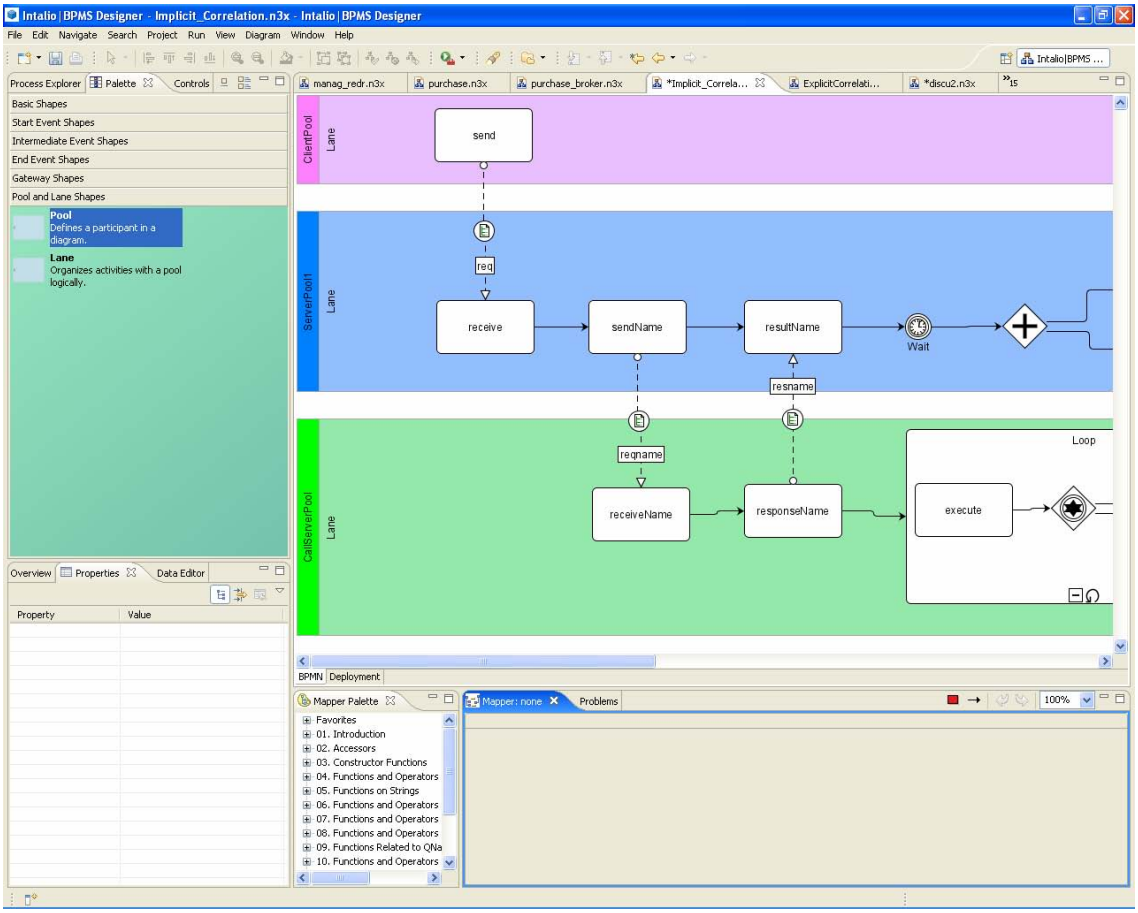


Figure 5. Intalio BPMS Designer.

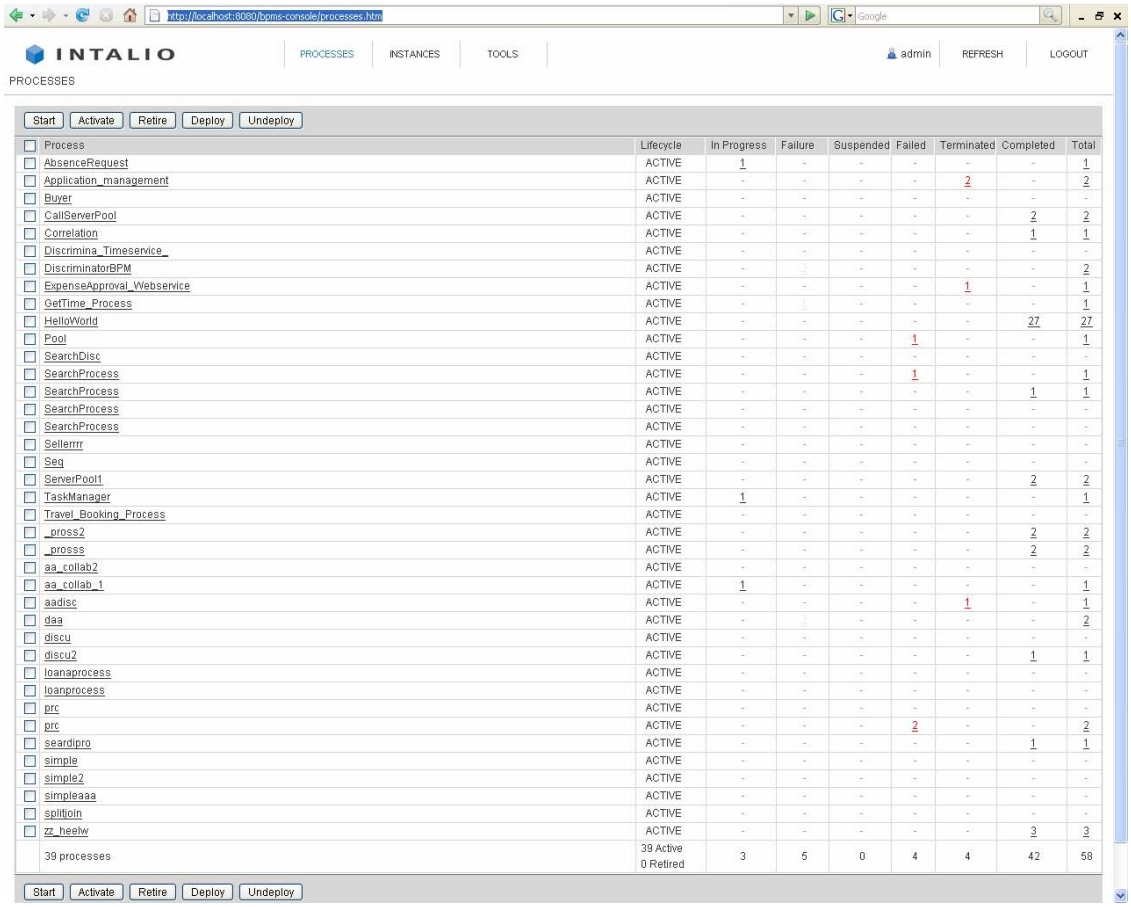


Figure 6. Intalio BPMS Server.

As BPMS products in general, Intalio BPMS Community edition is intended to cover all basic aspects of business process management including process modeling, deployment, execution and monitoring. The whole suite relies on BPMN as a visual notation and BPEL as an execution language. Intalio supports the WS-BPEL 2.0 specification which is claimed to be more portable than the earlier version. The WS-BPEL 2.0 descriptions which are generated based on BPMN can be deployed into the Server with one click in the Designer. The server also supports BPEL4WS 1.1 but for those files, a separate deployment package must be created. Other model formats were not supported by the used version. In the Designer, most of the BPMN 1.0 specification's elements are implemented. The few missing elements include *Data object* elements, *Cancel* elements and *Complex Gateways*. Additionally, Intalio BPMS 4.4 does not support abstract messaging between pools or merging between exception and normal flow.

Intalio is clearly focused on executable business processes based on Web Service technologies. It is not intended to be a tool for pure business analysts: basic understanding of the underlying technologies is definitely needed. Intalio Designer includes most of the BPMN elements relevant to executable business process models. In the Intalio Designer, external activities and message flows are mapped to specific

interface operations and message definitions using WSDL. The input and output message structures of the process are indicated with XML Schema elements. WSDL and XSD files are imported into the project space same way than any other files. Different elements of the documents can be browsed using the tree-like view of the file system. The service calls are modeled by introducing an additional *Pool* which contains the operations from the WSDL. The conditions are expressed using XPath. This is clearly an operation requiring technical expertise.

The private process then interacts with this external participant with message flows. Also the interface that is used to initiate the private process is expressed as a *Pool* which sends the message to the *Start event*. When the diagram elements are in place, the process data is mapped to Web Service inputs and outputs. For instance, it is defined how the eventual response message is constructed from different data sources.

Intalio Designer provides a graphical user interface for mapping variables with each other. The mapping tool provides a graphical interface for data combining and manipulating data. It eventually generates XML transformations and XPath expressions but these technologies do not have to be known by the process designer. Despite that, the mapping process is relatively complex and understanding of the Web Service technologies is needed. In addition to fully automated process constructs, the designer includes also tools for designing the user interfaces for human activities using, e.g., XForms.

After the process has been modeled and concrete services, messages and data have been defined, a corresponding BPEL description can be automatically generated in Intalio Designer. Also missing WSDL files are generated, including a description for the interface through which the process can be initiated. This executable process can be automatically deployed in Intalio BPM Server. In the server, the deployed processes can be started using a web-based form for giving the parameters for the initiative request document. The deployed processes are started and stopped through the web-based user interface of the server. The state and history of all deployed processes are also presented.

Intalio does not support any kind of round-trip engineering between BPMN and BPEL. After the BPEL file is generated, it can be modified only in XML format. A BPEL file cannot be imported into Intalio Designer and if the BPEL file is manually modified, the effects on the BPMN model are not indicated in any way. Intalio BPM Server presents the original BPMN model together with the executable process. In case of exceptions and faults, the related process activities are marked on the BPMN model. Otherwise the presented diagram is static.

5.2.2 eClarus Business Process Modeler for SOA Architects

At the time of this study, only a few tools supported both generating BPEL from BPMN and the other way around. One of these is eClarus Business Process Modeler for SOA Architects[6] which is the most advanced of the BPM products of eClarus. According to the web site, eClarus requires Windows XP or Linux. The tests were carried out in Windows environment. The tool includes only modeling and export/import capability, i.e., no execution engine is included, for instance. eClarus Business Process Modeler for SOA Architects is a commercial product built on Eclipse platform. The Eclipse foundation enables integrating eClarus easily with other Eclipse-based tools. This is important because of the restricted functionality provided by eClarus. According to the web site, the generated BPEL is compatible with the most significant commercial BPEL engines. Comparing to an integrated solution, eClarus does not include a BPEL engine, which makes the deployment of the generated BPEL descriptions more complicated. Due to the compatibility problems of BPEL4WS 1.1, one cannot be completely sure that the processes will be directly executable with any BPEL engine.

The modeling language used in eClarus is BPMN and it supports all the elements in the 1.0 specification. UML 2 Activity Diagrams created with IBM Rational Software Architect can also be imported but they are translated to BPMN. For this purpose, a proprietary format is used. Importantly, eClarus only supports the BPEL4WS 1.1 specification. This means that it is not possible to import into eClarus BPEL files generated with Intalio, for instance. As the name indicates, this eClarus product is also designed for modeling executable business processes. Compared to Intalio, however, some aspects of eClarus can be considered to improve the collaboration between business analysts, architects and designers. For instance, a change management tool is included to ease the comparison between *as-is* and *to-be* process models.

The basic modeling steps related to eClarus are to a large extent similar to the ones in Intalio but also differences occur. First, the BPMN elements and the connecting flows are created using an intuitive drag-and-drop user interface. Next, the technical details are added so that the executable code could be generated. This means defining the Web Service operations that carry out the activities, their message flows and conditions for the branches and merges in the control flow. The conditions for the flows are stated by XPath expressions which access the variable data in the messages. Also BPEL <partnerLink>-properties have to be filled in to indicate partners of the message. In contrast to Intalio, the WSDL operations or the initiating interface invocations are not presented by separate *Pools*. Instead, the services and their message flows are included in the properties of the activities which are accessible through separate menu. In other

[6] http://www.eclarus.com/products_soa.html.

words, in most cases message flow is not visible and it is not mapped to BPEL. It can be said that this simplifies the models: they include only the actual activities of the process and the service calls are hidden behind the scene. Information on the service implementing an activity is still closely connected to it, not dependent on a separate message flow. However, introducing the concept of a message flow inside a BPMN *Pool* may confuse some users. Web Service artifacts are imported into the eClarus file system similarly to Intalio, although with eClarus they are not project-specific. The basic modeling view of eClarus is shown in Figure 7.

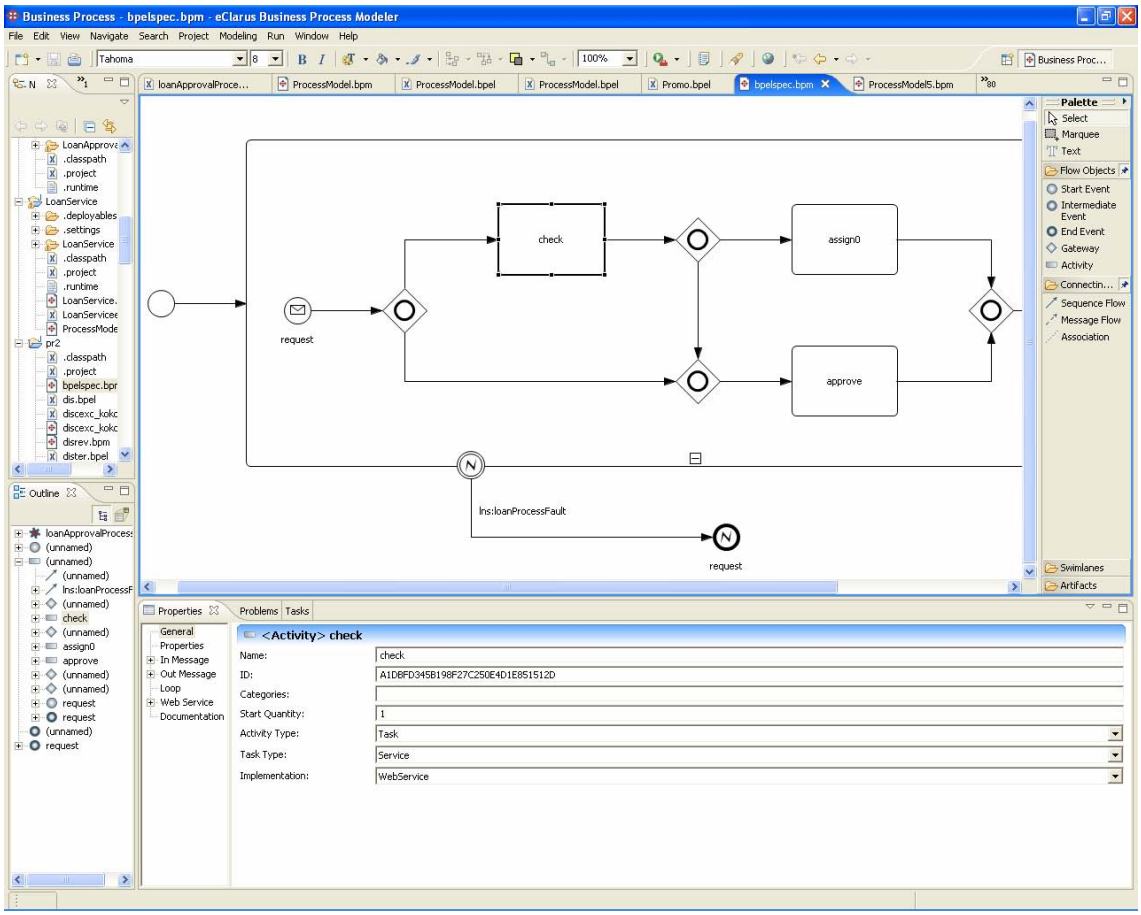


Figure 7. eClarus Business Process Modeler.

The most important functionality of eClarus is the BPMN-BPEL round-tripping, meaning that BPEL files generated based on BPMN models can be edited and then imported back to the tool to generate a diagram reflecting the changes. However, the basic functionality includes only importing of BPEL processes and generating the diagram based on that. This is indeed an effective way to keep the models consistent with the executable processes and to communicate with different people during the process lifecycle. However, there is no automatic way to ensure the correctness of the process behavior after modifying the BPEL model: this must be assessed by human experts. As the complexity of the models increases, this task becomes more difficult. It

can be said that with most real operational business processes, assessing whether the BPEL is functional is impossible without automated tool support. Therefore, both tools lack adequate support for process mining in which problems such as deadlocks or creation of multiple instances would be recognized. This kind of verification, in turn, requires a formal definition of the BPMN semantics.

Significant academic work has been quite recently conducted on this area (Dijkman et al. 2007), but it is likely to take some time before these results are fully exploited in the development of industry-strength tools.

5.2.3 Progress of the testing

We carried out six test cases in both tools. These test cases represented different kinds of business process models. The first test case was a very basic one and its purpose was to indicate the minimum amount of effort that these automatic transformations require. The next four test cases covered patterns that were found problematic in the existing research. The final test case represented a collaboration process. It was carried out in order to find out the limitations of BPEL in this area.

Because the test cases covered patterns (such as arbitrary cycles and multi-merges) that were considered difficult from the transformation point of view, the transformations did not always succeed. This was one of the goals of the testing – to determine the limitations of the state-of-the-art tools. In these cases, we sometimes tried to restructure the diagrams so that the same behavior could be captured without including patterns that were not supported by a tool. In many cases this was a success but there was often a trade-off included in, e.g., understandability of the diagram or in readability of BPEL code. Nevertheless, it can be said that even though not all diagrams could be transformed with existing tools, a workaround could often be formulated.

In the basic case and in cases involving parallel behavior or correlation (collaboration), so-called dummy Web Services were used to ensure that the execution actually corresponded to the modeled process. In other cases they were not considered necessary – the correctness of the BPEL process could be inspected by only reading the code. As was pointed out, this kind of analysis would not have been enough for more complex processes, but because the scope of the test cases was very limited, the inspection could be carried out. The results of the test cases were analyzed based on the following criteria:

- *Transformation success:* whether or not code could be generated based on the diagram. This also means that the generated code must correspond to the diagram.

- *Required manual operations:* what kind of modifications and elaborations were needed after creating the diagram to enable code generation. This includes all mappings to implementation details. Additionally, if the diagram needed to be modified to be able to carry out the transformation, i.e., the diagram could not be transformed as such, this is mentioned. Also at this point, the results are reported separately for each tool.
- *Occurred problems:* if the transformation was not successful, the reason for this is reported. This section is much related to the previous one because the problems were often solved through manual work.
- *Required knowledge:* what kind of knowledge other than familiarity with the tools was required to carry out the transformation? This part is also closely related to the manual work because these operations are based on specific knowledge about the technology or the limitations on the transformation support.
- *Quality:* after a successful transformation, the generated code was inspected in more detail, evaluating, e.g., readability of the code.

5.2.4 Test case descriptions

The first test case is the "Travel Booking process" (Figure 8). This is a simple process involving none of the patterns that are regarded especially problematic from the transformation point of view. However, in Intalio it was necessary to use a separate looping sub-process instead of a gateway. At the beginning of the process, the credit card information is checked. If this raises an error, the process is terminated and a separate error message is sent. The travel booking consists of reserving a flight, a hotel room and a car. Flight and hotel reservations always succeed but the car rental may require retries. When all the reservations have succeeded, a confirmation message is compiled and that message is sent as an output of the process. The process covers several basic elements and control flow aspects of BPMN. These include at least loops, parallel joins and exception flow. Therefore, the first test case covers the basic transformation capability of the tools with a relatively simple process. This test case was used to assess the basic quality of the generated code and to evaluate how much manual work and technological knowledge these transformations require at minimum. Therefore, all the activities were implemented as service invocations using so-called dummy services which correspond to the required interface but do not provide any real functionality.

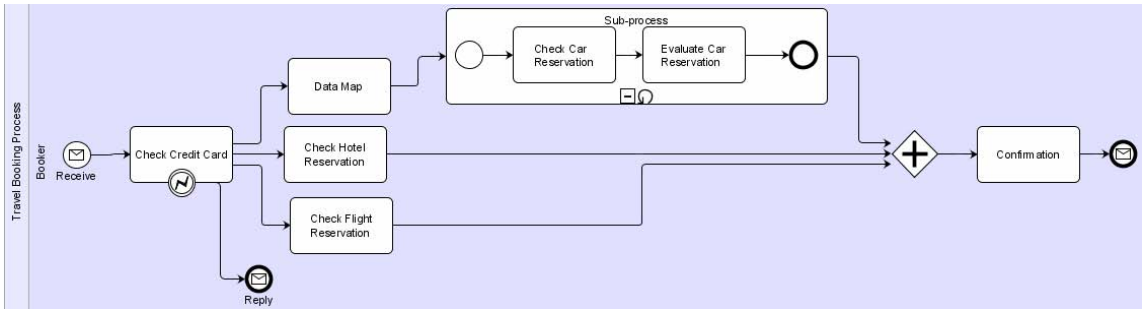


Figure 8. Travel booking process modeled with Intalio.

The second test case is the "Loan Approval process" (Figure 9) introduced in BPEL4WS 1.1 specification (Andrews et al. 2003). This process also does not involve complex task structures but it is still mentioned as a problematic case for BPMN-to-BPEL transformation by Gao (2006). Here, the exception handling of the original BPEL process was omitted to be able to point out problem areas in the transformations. Considering that BPMN should be able to present a visual notation for BPEL, being able to represent a process described in the BPEL specification can be considered important. The process is initiated with a message indicating the amount of loan requested and personal information for the applicant. If the requested amount is high enough, a separate approval service is invoked. If not, a risk assessment service is chosen to determine whether the applicant poses a low enough risk level. In this case, the application can be approved automatically. However, if a high risk level is indicated by the assessment service, the separate approval service must be invoked despite the low amount of the application. At this point, the control flow crosses paths with the high amount branch, resulting in invoking the same service. The flow tokens of the branches are mixed, indicating that the diagram is not BPEL-isomorphic and thus direct mapping cannot be carried out (Gao 2006, p. 5). This process was implemented using only internal data manipulation because the focus was on control flow, not on messaging between services.

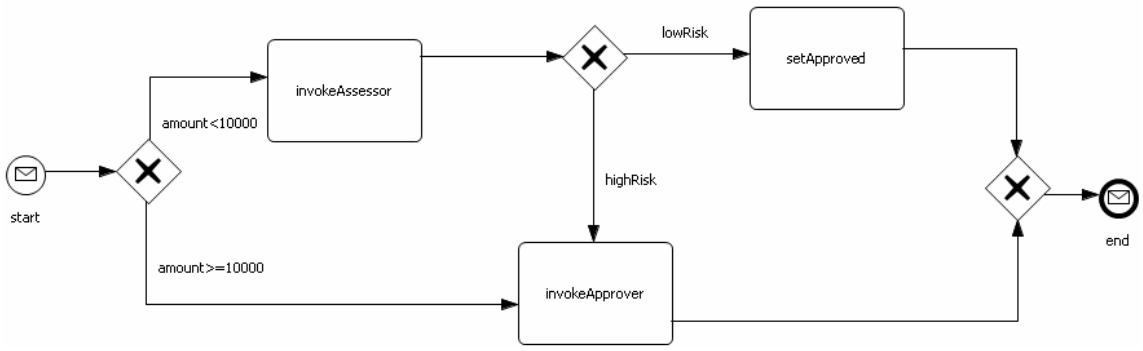


Figure 9. Loan approval process modeled with eClarus.

Next three test case processes include workflow patterns that have been found problematic from the transformation point of view. Test case number three (Figure 10)

introduces *Arbitrary cycles*, i.e., loops with more than one entry or exit points. The model is based on (White 2004b). The fourth test case (Figure 11) introduces the *Discriminator* pattern as described in the BPMN specification (Object Management Group 2006, p. 119) – the diagram includes a parallel split which has a corresponding exclusive merge. The process does not correspond to the *Discriminator* solution proposed by Workflow Patterns Initiative (2007), because they state that complex gateway should be used to be able to let only one token through the latter gateway. Nevertheless, the example process here can be considered problematic because the splitting and joining gateway do not match. In this process, two databases are queried in parallel. However, when either one of them responds, the result is sent to the client or the process is ended: the execution is not intended to stop and wait for the other activity (i.e., the other database query) to be completed as well. In this case, the actual service calls were carried out to ensure that the parallel activities do not introduce any new problems. The fifth test case (Figure 12) resembles the previous one. It includes the *Multi-merge* pattern in which parallel control flow branches reconverge without synchronization so that the activity following the implicit merge is executed once for each incoming branch. In the example process, application management consists of two parallel activities: application processing and auditing. These activities can be executed in parallel. They both require closing the case after their execution. The same *close case* activity should be executed separately for both processing and auditing. This is the point where the *Multi-merge* pattern appears. In this test case, only internal data manipulation was used due to the similarity with the fourth test case process.

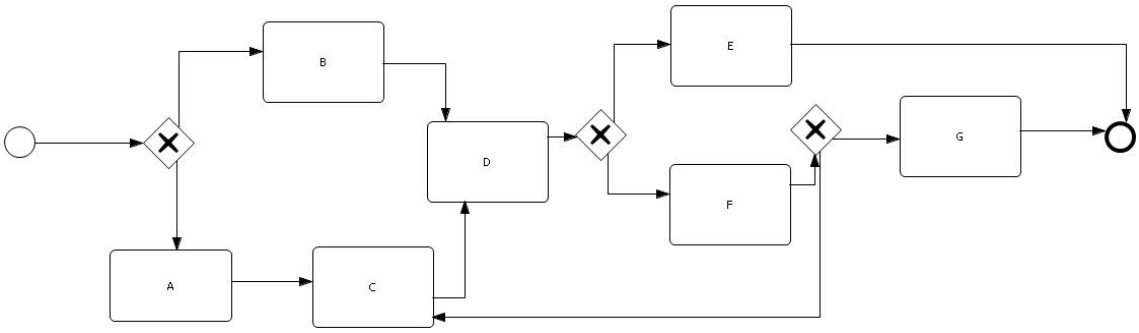


Figure 10. Arbitrary cycles modeled with eClarus.

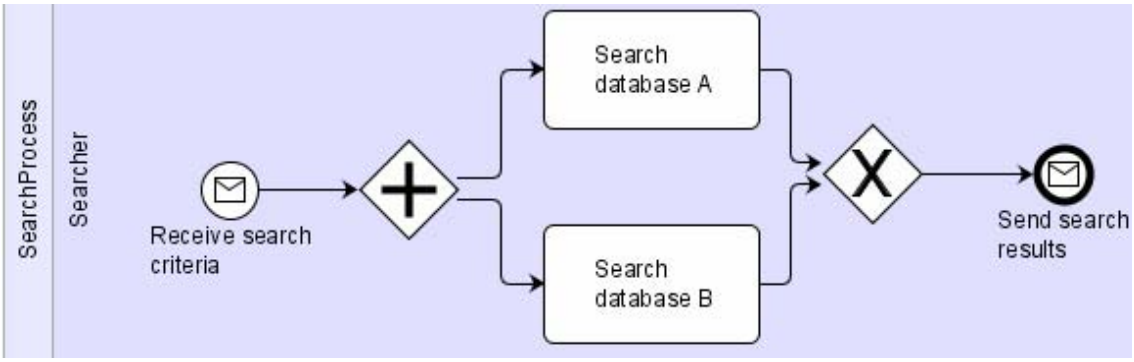


Figure 11. Parallel database search process modeled with Intalio.

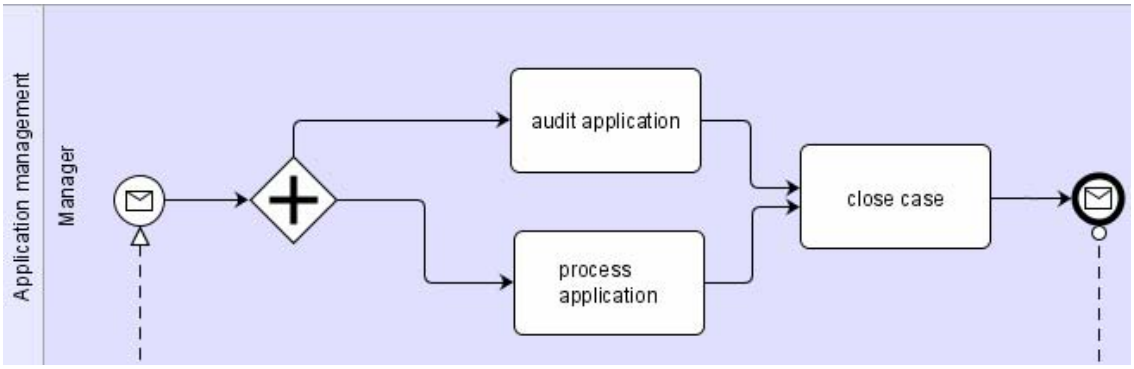


Figure 12. Application management process modeled with Intalio.

The final test case (Figure 13), "Purchase process" is a collaboration process. Essentially, the diagram involves two participants with their own processes, i.e., two Pools with separate sequence flows. It is not an abstract process because it includes the specific service calls involved in the process. However, this process only includes the points of interaction between the participants, not their internal business logic. These kinds of process descriptions can be related to, e.g., Partner Interface Processes (PIP) of RosettaNet. However, based on the presented diagram, an implementation with full details, such as WSDL descriptions are derived. In the process, the buyer first sends an order to the seller. Having accepted the order, the seller sends the goods which are received by the buyer. After that, the seller sends an invoice to the seller. This is followed by payment whose sending ends the process from the buyer's point of view and acceptance from the seller's point of view.

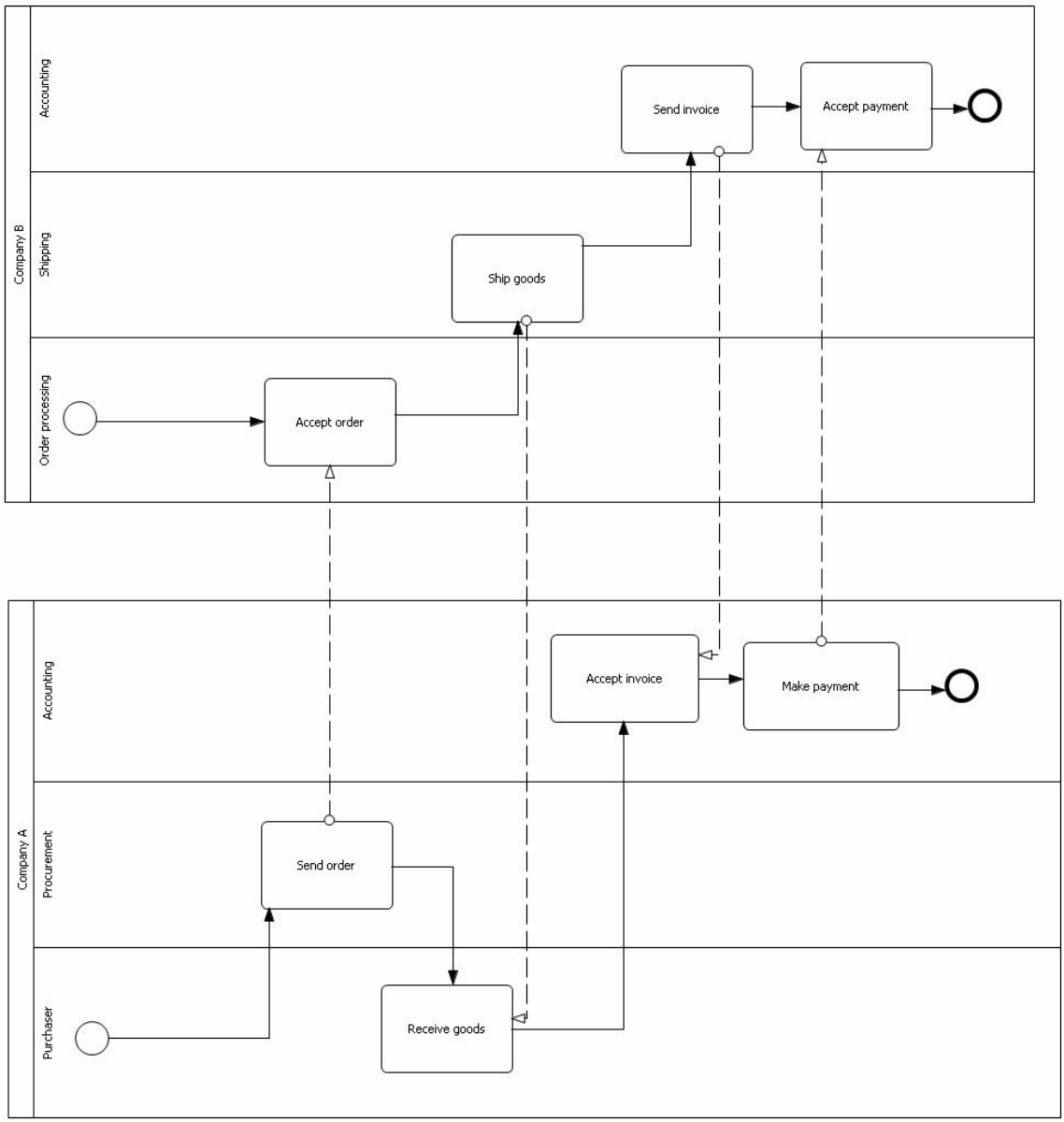


Figure 13. Purchase process modeled with eClarus.

6. Test case results

In general, it could be said that tool vendors have not been able to overcome all the fundamental limitations that apply to the transformations between graph-oriented and block-oriented languages. However, it should be pointed out that most of the test cases represent the most difficult situations – much can be done without any problems, but a fair amount of technical knowledge is required in any case.

6.1 Transformations

Both tools used in the transformations between BPMN and BPEL succeeded in the transformation of the first test case diagram. However, although the tools are based on the same notations, they impose different restrictions on the BPMN models and require different modeling techniques for the transformation to succeed. For instance, Intalio does not allow the use of loops which are based on *Gateways*. Instead, the separate *Looping subprocess* element must be used. It should be pointed out that subprocesses did not pose problems in this context even though no BPEL extensions were used. The looping subprocess mapped to BPEL <while> and the <scope> element was used to define the variables specific to that subprocess.

The methods for specifying these execution details are very much tool-dependent. This is understandable because unlike the notation, the methods for relating the diagram to e.g., variables are not standardized. The tools have different perspectives on processes, and their relationship to BPEL and Web Service elements. For example, eClarus requires that the user specifies the BPEL <partnerLink> elements related to each Web Service activity in order to determine which service interactions are parts of the same message exchange. In Intalio, this is deduced from the message flows between the *Pool* that represent the executable process and the *Pools* that represent the external Web Services. It should be pointed out that even though the tools would succeed in generating BPEL, additional procedures are required to be able to execute the process. These are also tool-dependent. For example in Intalio, the variables have to be initialized. Without this, the execution of the BPEL process will crash, at least in the Intalio's own execution engine and Geronimo application server.

The BPMN-to-BPEL transformation of the "Loan approval" process was successful with eClarus but it failed with Intalio. The error message indicated that there were conflicting flows from the upper fork to the lower merge. The third test case failed with both of the tools. With Intalio, it is not even possible to draw arbitrary cycles because the looping sub-process element must be used. With eClarus, the diagram could be drawn but BPEL could not be generated: the looping to a branch on activity C was not

supported by the BPMN-to-BPEL mapping. The fourth test case was also a failure with both tools. Notably, eClarus recognized the pattern and explicitly stated that it is not supported by the mapping. More importantly, however, Intalio generated incorrect BPEL code based on the diagram. In the used version of the Intalio Designer, the type of the merging gateway was not taken into account in the transformation: the generated BPEL was executed as if there had been a synchronizing parallel gateway instead of an exclusive one.

Also the transformation of the fifth test case model failed with both tools. With eClarus, no BPEL code could be generated. The error message stated that the uncontrolled flows at the activity following the merge were not supported by the mapping. With Intalio, the results resembled the previous test case: BPEL was generated but the execution behavior was incorrect because a synchronizing merge was assumed. Therefore, the *Close case* activity was executed only once instead of twice. However, this model could be restructured so that the transformation could be carried out. This was done by replicating the *Close case* activity so that it appears after both *Audit application* and *Process application* activities. This means that both activities are mapped to the same Web Service. The solution results in multiple instances of the *Close case* activity, but it would not have been possible to avoid that due to the nature of the *Multi-merge* pattern.

Due to the focus of SOAMeS, the final test case requires special attention. Both tools were able to generate BPEL based on the diagrams. However, at least in the case of eClarus, the BPEL code does not correspond to the diagrams as such. The tool only allows generating BPEL for one *Pool* at the time. In the BPEL code, the other pool and the message flows are completely ignored and the Web Service information is attached to each *Activity* separately, as described earlier. This means that the collaboration process cannot be maintained in one place and that the interactions cannot be visualized. The other process must be modeled separately. Moreover, eClarus only generates BPEL based on the processes, not WSDL files which would be necessary to enable invocations among processes. The approach of Intalio is more feasible. It treats the *Pools* as separate executable processes and automatically generates BPEL and WSDL files for both of them. This way the initiated Buyer process invokes the Seller process which is then automatically started in the BPMS engine.

The BPMS presents the status of both processes, pointing out the activity in the collaboration diagram that is currently in progress for each process instance. This means that the collaboration process could be realized by deploying two interaction processes for both participants. Internally, the participants could map the process activities to more detailed processes indicating the business logic that they do not want to share with their business partners. For more complex collaboration processes, the messages need to be routed to correct process instances by explicitly defining BPEL correlation sets. This functionality is provided by both of the tested tools. Notably, defining correlation sets is a complex task requiring a substantial amount of BPEL knowledge.

The use of abstract BPEL would be feasible in collaborative processes such as the last test case process. The tested tools did not provide any support for generating abstract BPEL based on BPMN, however. If abstract BPEL was used, one difference to the approach used in the last test case would be that it could be more accurately defined, which parts of the participants' internal process behavior would be hidden. Following the example in BPEL4WS 1.1 specification (Andrews et al. 2003, p. 9), one could define in the abstract process that the seller's process includes a decision point realized in BPEL <switch>. However, the conditions for the decision making could be hidden, i.e., opaque. This way, the alternatives for the behavior could still be observed, as opposed to the situation in which all but the outcome would be hidden.

6.2 Code quality

In the first test case, the quality of the resulting BPEL code is high for both tools. The strict conformance to BPEL specification was not evaluated but it can be said that no questionable elements which would affect the execution could be found. The BPMN-to-BPEL mappings correspond to a large extent to the basic mappings presented in the literature. This means that the structured BPMN elements map to BPEL <sequence>, <flow> and <while>, for instance.

The condition related to the handling of credit card error is expressed with BPEL <switch> or <if> element, depending on the BPEL version used. The Web Service calls related to the BPMN activities is indicated with BPEL <invoke> and <reply> elements and data manipulation with <assign> and <copy>, for instance. In general, the resulting BPEL code is concise: no unnecessary constructs could be recognized. As claimed, with eClarus it is possible to import the BPEL files so that the possible modifications made to the code are indicated in the diagram and the execution-specific information can be derived from the BPEL file. However, because BPEL does not store diagram information, the BPMN elements are laid out in a quite disordered manner. The resulting diagram is based directly on executable elements. This means that on one hand, additional information such as text annotations is lost. On the other hand, extra elements are added to reflect the actual BPEL process. These include, e.g., empty activities which are not necessary in BPMN diagrams. Additionally, the software is not aware of whether or not the BPEL process is based on a diagram previously created with eClarus. To achieve better maintainability of the processes, the use of change tracking feature must be applied or eClarus must be integrated with a more sophisticated version management system.

In the case of Intalio, the incorrectness of the generated BPEL in test cases four and five can be seen as a serious quality issue. The tool should not be able to generate BPEL

because unstructured split/join-patterns are not supported by the mapping. In the final collaboration process, the only quality issue is that the processes of different participants are presented completely separately from each other. This is of course based on the nature of BPEL in general but if the processes are generated with the same tool at the same time, for example a comment field indicating this relationship would have been beneficial.

As with any programming language, the generated BPEL code needs to be thoroughly tested before executing it in a production environment. It can be said that one cannot yet completely rely on the code that these tools generate. However, it should be borne in mind that many test cases highlight the weaknesses in the transformations. Therefore, an experienced process modeler might be suspicious about the transformation from the beginning and test the executable process with additional caution.

The test case results are summarized separately for Intalio in Table 5 and for eClarus in Table 6. The test cases were kept relatively simple in order to clearly point out the shortcomings in the transformations. Because of this, however, it cannot be clearly said whether one tool is superior to the other. As was stated earlier, the purpose of this study was not to provide recommendations for purchasing products, but to demonstrate the what kind of limitations could apply in currently available tools, and to outline what kind of value these tools and technologies provide today. Based on the results, it can be concluded that the differences between the tools were not significant. This is considered to be based on the fact that they were based on standards, BPMN and BPEL, not proprietary solutions. The differences on the test results are likely to stem from different internal transformation algorithms of the tools. Additionally, the tools pose different restrictions on the models that can be transformed to BPEL. For example, eClarus requires that *Gateways* are used for merging instead of uncontrolled flow. Finally, the tools use different versions of BPEL. BPEL4WS1.1 does not pose as strong restrictions on control links than WS-BPEL 2.0, which can be predicted to result in more successful transformations. The conducted tests do not bring this up, however.

Table 5. Test case results for Intalio.

	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case 5**	**Case 6**
Name	Basic	Mixed tokens	Arbitrary cycles	BPMN Discriminator	Multi-merge	Collaboration
Success	yes	no	no	no	no	yes
Manual operations	data mapping, forking conditions, variable initialization	see case 1	n/a	n/a	n/a	see case 1; correlation, defining which is the executable process
Problems	execution crash without variable initialization, enforced subprocess	conflicting flows	loops must be encapsulated in subprocesses	invalid BPEL generated	see case 3	two BPEL processes from different perspectives
Knowledge	existing services, tool principles, BPEL basics, XPath basics	see case 1	n/a	knowledge of invalid BPEL mapping	see case 4	knowledge of BPEL's relationship to collaboration, understanding correlation, knowing the business partner and its relevant IT assets
Code quality	OK	n/a	n/a	n/a	n/a	no indication that the processes relate to one another

Table 6. Test case results for eClarus.

	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case 5**	**Case 6**
Name	Basic	Mixed tokens	Arbitrary cycles	BPMN Discrimi-nator	Multi-merge	Collaboration
Success	yes	yes	no	no	no	yes
Manual operations	data mapping, forking conditions, partnerLink specification	see case 1	n/a	n/a	n/a	see case 1; correlation, defining which is the executable process
Problems	none	none	lack of uncontrolled flow support	invalid BPEL generated	unsupported pattern recognized	two BPEL processes from different perspectives
Knowledge	existing services, tool principles, BPEL basics	see case 1	n/a	knowledge of pattern terminology	see case 4	knowledge of BPEL's relationship to collaboration, understanding correlation, knowing the business partner and its relevant IT assets
Code quality	OK	OK	n/a	n/a	n/a	no indication that the processes relate to one another

7. Conclusions

In this research, we have studied and evaluated the state-of-the-art in business process modeling and execution technologies. The research included a literature study on relevant languages as well as tool-based testing whose purpose was to find out to what extent the transformations between business process modeling and execution languages can be automated. To reveal the limitations of the tools, the test cases included process patterns that were considered challenging in the existing research. In addition to data indicating success or failure, the results cover additional requirements that the use of these tools pose. This means that we have outlined the phases in the modeling process and the different competences required. However, thorough formulation and evaluation of such conclusions would have required case studies in real enterprise environments. These were out of the scope of the research at this point.

7.1 Business process modeling and execution languages

Despite the large variety of languages in the Business Process Management domain, the most prominent alternatives in the categories of both process modeling and execution can be recognized. BPMN has been designed with only business processes in mind. It can be used to model executable processes but it is also easy to understand by people who are familiar with, e.g., Web Services. Additionally, processes can be modeled at a higher level of abstraction to communicate with business experts. BPMN has a close relationship with executable languages, namely BPEL. This translation has also been widely implemented in tools, although some restrictions exist. UML 2.0 Activity Diagram is also suitable for business process modeling but BPMN can be considered more relevant, especially when it comes to tools with translation support.

One clear weakness of BPMN is the lack of widely supported model interchange format. Currently, editing the same models in different tools will be very difficult if not impossible. BPDM or XPDL may provide an answer to this problem, but they are not widely supported by tools. This field should be further investigated in the future.

In the category of executable business process languages, BPEL does not have serious direct competitors. Unfortunately, BPEL has its shortcomings related to complexity, abstract processes, portability and human activities. It can be said that rather than using BPEL extensions, these problems can be overcome by generating BPEL code automatically whenever possible and restricting its use to private orchestrations. This means using other languages in, e.g., process modeling (BPMN or UML AD), interchange (XPDL or BPDM, if tools support these) and choreography design (e.g., WS-CDL).

7.2 BPMN and BPEL tools – provided value and limitations

Based on the results, it can be said that the transformation between BPMN and BPEL is well supported by the available tools. Because BPEL lacks standardized visual notation, BPMN modelers with BPEL export support are, hence, a good substitute for pure BPEL modelers. This is already a step forward because BPMN will most likely gain popularity in the near future.

Some features of the tools indicate that the only purpose of BPMN is to visualize executable BPEL processes. However, visualizing BPEL is only one of the objectives of BPMN and Web Service based execution only one objective of Business Process Management. In addition to the basis for execution, process models can also be used to facilitate communication in the earlier stages of the development. It would be beneficial if the models on different levels were not completely separate from each other. The tools should thus distinguish more clearly on different levels of validity related to Business Process Diagrams. Indeed, it should be made sure that the processes can actually be executed before generating BPEL but it should also be possible to first use the intuitive modeling elements of BPMN more freely and make the necessary restructurings at the point when the BPEL is generated.

While using BPMN with necessary additional execution information to model BPEL processes is justified, the experiments also showed that modeling executable processes is not a task of a business analyst but technical competence is required. The experiments clearly indicate that even if the modeling is carried out with appropriate tools and technologies, the path from the business process diagrams to process execution consists of a wide range of different kinds of tasks which require different kind of knowledge, so Business Process Management clearly requires that people with different responsibilities are included in the project. There is a need for IT-people with specific knowledge business process modeling and execution – which is a domain of business informatics.

In order to truly benefit from business process modeling, the tools must be accessible by both business and IT users. Based on the tests, it is concluded that the tools do not fully meet this requirement. In the tested toolset, the following shortcomings were recognized:

- *No alternative views for the model:* there are no mechanisms for applying different modeling restrictions on different levels (e.g., business-oriented vs. technical level) and moving between these levels in a systematic way. For example the business users might not be interested the detailed Web Service information of the activities. On the other hand, for them it would be beneficial to clearly indicate the responsible business partner for each activity.

- *No feasible support for model interchange:* the tools do not support languages which could be used to export the diagrams with necessary execution details so that they could be used in different tools. One reason for this is that the standards for model interchange are currently at an immature stage. Possible languages for this purpose include XPDL 2.0 or BPDM, but they were not studied here due to the lack of adequate tool support.
- *Limited support for checking the correctness of the model:* the tools check that the BPMN model is valid in the sense that a BPEL file can be generated, but there is no way to automatically ensure that the process terminates correctly, meaning that the process is free of deadlocks and livelocks, for example. With complex real life business processes, ensuring the correctness of the model is a very difficult task, even for a process modeling expert. Therefore, tool support for this kind of analysis would be very important.
- *Limited collaboration functionality:* besides the change tracking of eClarus, no support for distributed collaboration is provided. Due to the nature of the domain, these kinds of features would be beneficial. Additionally, no support for modeling collaborative processes with business partners is provided. For instance, there are no mechanisms for sharing business models so that only limited visibility on details is granted. The technologies do not provide enough support for this, though, and therefore no tools with such functionality were found in the preliminary study on available products.
- *Different modeling approaches in different tools that use the same notation:* BPMN specifies the elements and the behavior they indicate but not how these elements should be used to model the process execution. The appearance of equal diagrams is substantially different.
- *Limited feasibility of the round-trip engineering feature:* the other tool introduces round-trip engineering but it can be said that the independent import and export features are not adequate for keeping the model and the executable process consistent in iterative development. This is because importing can introduce unnecessary changes to the model. The appropriate mechanism would be to merge the changes in the BPEL file into the model, not to rely on imported BPEL code alone.
- *Limited support for managing existing service assets:* WSDL and XML Schema files can be imported and used in the model but there are no mechanisms for finding appropriate assets. A versatile service repository must not necessarily be a part of a BPMN modeler or even a BPMS. It should still be pointed out that in order to really leverage from business process models, everybody who takes part in to the modeling should have at least a preparatory view on what existing services and data structures will be used. This means that at least the existing assets should reside on a repository accessible by all contributors. The proper use of metadata would also be advisable.

- *Limited support for diagram restructuring to enable execution:* besides the limited internal restructuring provided by eClarus, the transformations often fail if the diagram cannot be directly mapped to BPEL. A limited amount of information is provided on the reason of the failure and no suggestions for overcoming the problem are given.

Based on the above, it can be said that reviewed tools and technologies have not yet fully matured. They are not yet completely able to fulfill the promises made by tool vendors or standardization organizations in the industry. This finding cannot be fully generalized due to the limited scope of the tool study, but because many shortcomings were based on the weaknesses in standards (such as gaps in expressive power and ambiguities in specifications, the research was not tool-specific. Technologies and tools develop rapidly, however: for example, during this study, new versions of the tools have been released and the standardization process of WS-BPEL has been completed. Given the current state of the tools, the executable business process models will for the most part be developed by IT specialists. However, the modeling tools enable business and IT people to collaborate on the same models which are based on the same notation. This prevents the realized processes from drifting too far from the business requirements they are meant to fulfill. Additionally, the productivity of the IT organization will rise, because the developers are not required to manually write the executable code on a detailed level.

Due to the close relationship between business processes and Service-Oriented Architecture, process modeling is an important step in the adoption of SOA in an enterprise. The objective of the modeling should be to design processes that could be eventually executed in the service-oriented environment. This means that appropriate technologies and tools should be used as much as possible, even if automatic code generation would not at first be the main objective of the modeling. As the adoption of SOA in the enterprise proceeds, the models can be gradually elaborated to enable execution.

Adopting a Business Process Management approach in an enterprise is not a task that could be carried out at once, even if a solid base for SOA already existed. Gradual adoption would be advised. For example, one could start by introducing the tools and technologies to the IT organization. This could be done with the freely available tools. If the enterprise has already developed a substantial amount of BPEL processes, testing a round-tripping tool would be advisable to evaluate how well the existing processes could be integrated with the modeling environment. Because the tools have not yet fully matured, this kind of preliminary evaluation is advisable before substantial investments are made. The next stage would be to determine how to organize the business process modeling in the enterprise and assign roles to different people involved. At this stage,

also the business people should be introduced with the used modeling tools. The required knowledge depends on how the roles are assigned, but it could be said that also the business users should be familiar with at least the basics of the modeling language.

7.3 Guidelines for tool selection

Because this report only covered two tools supporting BPMN and BPEL, we do not provide you with recommendations about which tool to purchase. Based on the reported test cases, one should have a better view on what to expect from the available tools today. One reason for not covering a wide range of products in our tests was that organizations have very different needs considering these kinds of products. Business process modeling and execution will not take place in isolation: automated business processes will be to a large extent based on existing IT assets. The modeling and execution tools also need to interoperate with systems that are already in use in the enterprise. This does not necessarily mean that the tool selection would be based on standards evaluations such as the ones discussed in this report. Using a proprietary solution might be a natural choice, especially if the company has considerable experience on certain vendor's products.

When a company has reached the state of selecting individual (possibly proprietary) products, it is beneficial to study the product evaluations conducted by the Workflow Patterns Initiative – they can be freely accessed on the website[7]. The evaluations provide an extensive amount of information about the patterns that the products' execution environments support. Due to the academic nature of the initiative, the evaluations are considered extremely reliable.

If open standards are used, one may refer to the standards' evaluations. In this case, it is necessary to take both the modeling and the execution standard into account and focus on the differences on their pattern support. An example of this kind of analysis can be found on Section 3 of this report. However, due to the differences in the model transformation methods and algorithms in different products and possible errors in the implementations of, e.g., BPMN specification, pattern-based standard's evaluation does not guarantee that any tool supporting certain modeling and execution standards would have the expressive power indicated by the standards' workflow pattern support. Tool-specific shortcomings may exist, but workflow patterns provide a valuable framework for identifying them, nevertheless.

[7] http://www.workflowpatterns.com/.

The most important product aspects are highly context-specific and discussing all of them would have been out of the scope of this research. However, based on our experiences, a set of guidelines for selecting modeling and execution tools can be presented. Using these, the most appropriate tool can be selected based on product documentation or evaluation versions of the tools. People from different positions in the enterprise should be involved in selecting the products, because business process management will have long-term effects on many levels of the organization. One option is to organize a steering committee to supervise the investments in BPM tools. In the tool selection, the following questions should be taken into account:

- *The role of business process modeling in the organization:* in this report, we have focused on business process modeling which aims at generating executable code from the models. However, in many cases business process diagrams act more as a tool for communication than the basis for execution. The advent of SOA and related technologies does not make this approach less relevant. Modeling languages such as BPMN are significant also in this context. If they are used instead of proprietary notations, the meaning of the modeling elements is more accurately specified. If one decides to define executable models in the future, the work does not need to be started from scratch. However, the role of the models should be taken into account in the tool selection. If a modeling notation is used also for less formal purposes, one should prefer tools that do not take the restrictions of the execution environment into account in the modeling phase. For example, Intalio BPMS would not be very suitable for such purposes.
- *Business process modeling tools already in use in the organization:* if processes are already being modeled in the organization using well-established tools or technologies, it is not obvious that one should move to using BPMN, for instance. However, if the current tools do not fully meet the modeling needs of the organization and, e.g., adopting BPMN is considered, different approaches for carrying out this shift can be taken:
  - *Examine the future directions of the currently used tools:* it seems that many important modeling tool vendors will begin to support BPMN in the upcoming versions of their tools. For example, QPR[8] has announced that the upcoming version of their BPM software will include support for BPMN and BPEL (QPR Software 2007).
  - *Use new modeling language concurrently with the old one:* many tools which previously have only used proprietary notations are now also provided with at least limited support or extensions for BPMN. These include at least IDS-Scheer Aris and Microsoft Visio. Although these cannot necessarily be

[8] http://www.qpr.com/.

considered as full-blown BPMN tools, they provide a good opportunity to experiment with the new modeling language and to determine how it would position in the process modeling traditions of the organization.

  - *Abandon the old tools and technologies:* if the existing process models are not directly connected to the execution environment, it may not be necessary to support the modeling tools they were created with. They can be recreated with the new language if necessary. The downside of this approach is that people have to be trained to use the new modeling language.

- *Business process execution tools already in use in the organization:* for example BPEL engines may have already been adopted in the enterprise. These are complex environments and replacing the old engine or using several engines concurrently is not necessarily advisable. In these cases, one probably should not invest in a BPMS. Additionally, one should make sure that the code generated by the modeling tool is compatible with the used execution engine. The processes must be considered more important than the tools. If the execution engine does not execute the standard processes created elsewhere as intended, it is definitely advisable to replace the old execution engine with a product that implements the standard more rigorously.
- *Integration needs:* business process execution engines rely heavily on, e.g., application servers and databases. Connectors to other enterprise systems, such as SAP are also sometimes included in commercial BPM products (e.g., Intalio's commercial version). These kinds of systems are often already in use in the organization. Hence, choosing an execution engine that is compatible with the already adopted products is advisable.
- *Willingness to invest in BPM technology:* adopting SA-based BPM can be a significant investment. For example, a license to eClarus Business Process Modeler for SOA Architects costs $1340 per unit (eClarus Software 2006) and to really benefit from the product, user training is also needed. It could be said that heavy investments in BPM products should be aligned with the organization's overall IT strategy. This means that the tools are purchased to support the organization's process and service orientation efforts. For more light-weight experimentations, freely available tools are recommended.

7.4 An outlook to the future

It does not seem likely that any modeling or execution language will make BPMN or BPEL obsolete in the near future. BPMN and its most serious competitor, UML, are now being driven by the same standardization organization which may indicate that a more clear distinction between the languages will be made in the future, i.e., that BPMN

will be used in business process modeling and UML AD in software engineering domain. In any case, it should be borne in mind that neither BPMN nor BPEL are actual e-business standards that the business partners should necessarily agree on. When choosing a standard, a company should take into account the language's suitability for the organization's purposes. This means that it should be evaluated whether the language is able to satisfyingly express the kind of processes that are developed in the organization. This also means that the language has adequate tool support and penetration so that full-scale adoption can be carried out. In most organizations, this makes academic languages such as Yet Another Workflow Language (YAWL) unadoptable, although they often have better expressive power than industry standard languages.

Although the languages are not likely to disappear, it is possible that the way they are used will change in the future. According to the MDA camp (Kleppe et al. 2003), in the future, the typical software development process will be largely tool-driven and most people will work on models on different levels instead of writing code. There will be an additional group of people, mostly working for the tool vendors, that defines the transformations between platform independent models, platform specific models and code. This is far from being realized, but it is quite obvious that the level of abstraction will rise – this has been a trend in computer science for a long time.

Automatic transformations promise to bridge the gap between business and IT and increase the productivity of the IT organization. The downside of this direction is the increased power of tool vendors. Enterprises become more dependent on the languages the vendors support and the transformations they implement. This means that if an organization has adopted a certain modeling language, they cannot easily change their execution language if the tool vendor does not provide transformations for this purpose and vice versa. In this scenario, open standards have a pivotal role. Although transformations can still be proprietary, open standards ensure that knowledge transfer will occur in, e.g., standardization organizations and scientific communities. Open source and freely available software will also have an important role, at least in the early stages of this domain. For example, an enterprise could evaluate the feasibility of a BPMS in general using Intalio. A commercial product could then be purchased later on if this is necessary due to the integration needs of the enterprise. Also Eclipse BPMN Modeler[9], an open source BPMN tool, is under constant development and its status should be monitored in the future. It was not included in this survey, because it currently provides only BPMN modeling functionality without any support for executable languages.

[9] http://www.eclipse.org/stp/bpmn/.

With the increased automation of business processes and the rise of e-business, also the business users need to be on some level familiar with the technologies that realize these processes. Fortunately, the learning curve is made gentler by tools which raise the level of abstraction above the technical details. It can be said that these tools will develop rapidly in the near future and careful first steps in their adoption can be already taken today. Companies which have already launched their SOA development can already start experimenting with the tools. As the organization has familiarized itself with the tools and technologies, they will bring a substantial amount of rigor in business process modeling and make the development work more efficient. Other companies could start by building a solid foundation to enable the execution of the processes. In a few years, the tools and technologies are likely to develop so that the full potential of business process modeling can be utilized.

8. Summary

Service-Oriented Architecture (SOA) has brought IT assets closer to the business process they are meant to serve. Therefore, visual process modeling languages and XML-based languages that control the process execution have recently become hot topics in IT. To keep the process models and the executable processes aligned, it would be ideal if executable code could be automatically generated from the models. We examined current business process modeling and execution technologies and tools and evaluated how well they fulfill this promise.

For business process modeling, two prominent language standards exist: Business Process Modeling Notation (BPMN) and UML Activity Diagram (UML AD). We considered BPMN the more promising one mainly due to its extensive tool support. For business process execution, Business Process Execution Language (BPEL) is the only relevant option. Essentially, BPEL is a high-level programming language with strictly restricted structure which is very much different from business process diagrams. The expressive power of BPEL is thus not as good as that of modeling languages. Because of this, certain restrictions recur in the transformation methods presented in the academia. We also found out that certain modeling constructs have ambiguous definitions in the specifications, which can lead to tool-specific problems in transformations.

In our research, we tested the currently available tools to evaluate their feasibility. We found out that the basic BPMN diagrams could be transformed automatically to BPEL, but this required a substantial amount of technical knowledge. In some more complex cases, the transformation did not succeed. Sometimes workarounds to these shortcomings could be built, but this required in-depth knowledge of the execution environment. In addition to transformation weaknesses, the tools also lacked adequate support for the collaboration between IT and business users and process modeling among business partners. The purpose of the tests was to provide guidelines for selecting Business Process Management products and identified their typical limitations, not to recommend specific products. Things that should be taken into account include existing modeling and execution tools and technologies, integration needs and willingness to invest in BPM.

Although the tools and technologies have not yet fully matured, we feel that companies that are moving towards Service-Oriented Architecture and can begin to gradually adopt them. It is likely that many of the current problems will be overcome in a few years. Business and IT users will be able to systematically collaborate on the same models and the developers do not have to constantly concern the details of the execution language. This will bring business and IT closer together and increase the productivity of the IT organization.

Acknowledgements

The authors would like to thank all members of the SOAMeS project groups at VTT and University of Helsinki for providing comments on this report. Especially we would like to thank Alexander Norta, whose insight on the topic was highly valued.

This research was funded by Tekes - Finnish Funding Agency for Technology and Innovation, VTT and the following Finnish companies: Elisa Oyj, Kesko Oyj, Metsäteho Oy and TietoEnator Processing & Network Oy.

References

Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D. & Thatte, S. 2003. Business Process Execution Language for Web Services (BPEL4WS). IBM, BEA Systems, Microsoft, SAP AG, Siebel Systems. http://www-128.ibm.com/developerworks/library/specification/ws-bpel/.

Dijkman, R.M., Dumas, M. & Ouyang, C. 2007. Formal Semantics and Automated Analysis of BPMN Process Models. Queensland University of Technology. http://eprints.qut.edu.au/archive/00007115/.

eClarus Software 2006, n.d.-last update. eClarus Business Process Modeler Order Form [Homepage of eClarus Software], [Online]. Available: http://www.eclarus.com/purchase.html [2007, 08/01].

Emig, C., Weisser, J. & Abeck, S. 2006. Development of SOA-Based Software Systems – an Evolutionary Programming Approach. International Conference on Internet and Web Applications and Services ICI. Vol. 6.

Gao, Y. 2006. BPMN-BPEL Transformation and Round Trip Engineering. eClarus Software (online). http://www.eclarus.com/pdf/BPMN_BPEL_Mapping.pdf.

Hill, J. & Drakos, N. 2006. Intalio Boosts BPMS Usage With Open-Source-Like License. Gatner Research. http://www.gartner.com/resources/145400/145460/intalio_boosts_bpms_usage_wi_145460.pdf.

Kanniainen, J. & Haajanen, J. 2007. BPEL Engines. State-of-the-Art Survey for SOAMeS-project. VTT Working Papers 69. VTT, Espoo. 30 p. ISBN 978-951-38-6620-4. http://www.vtt.fi/inf/pdf/workingpapers/2007/W69.pdf.

Kavantzas, N., Olsson, G., Mischkinsky, J. & Chapman, M. 2004. Web Services Choreography Description Language (WS-CDL) 1.0. The World Wide Web Consortium. http://www.w3.org/TR/ws-cdl-10.

Kleppe, A.G., Bast, W. & Warmer, J.B. 2003. MDA Explained: The Model Driven Architecture: Practice and Promise. 1st edn, Addison-Wesley Professional.

List, B. & Korherr, B. 2006. An evaluation of conceptual business process modelling languages. Proceedings of the 2006 ACM symposium on Applied computing. Pp. 1532–1539.

Melenovsky, M. & Sinur, J. 2006. Having a BPM Maturity Model is Important for Long-Lasing BPM Success. Gartner, Business Integration Journal, Nov-Dec 2006. http://www.bijonline.com/index.cfm?section=article&aid=797%20.

Mendling, J., Lassen, K. & Zdun, U. 2006. Transformation Strategies between Block-Oriented and Graph-Oriented Process Modelling Languages. In: Multikonferenz Wirtschaftsinformatik, eds. F. Lehner, H. Nösekabel & P. Kleinschmidt. GITO-Verlag, Berlin. Pp. 297–312.

Mendling, J., Neumann, G. & Nuttgens, M. 2004. A Comparison of XML Interchange Formats for Business Process Modelling. In: EMISA2004, Informationssysteme im E-Business und E-Government, eds. F. Feltz, A. Oberweis & B. Otjacques. Vol. 56 of Lecture Notes in Informatics (LNI). Lucembourg. Pp. 129–140.

Miguel, A. 2006. April 19th 2006–last update, WS-BPEL 2.0 Tutorial [Homepage of The Eclipse Foundation], [Online]. Available: http://www.eclipse.org/stp/b2j/docs/tutorials/wsbpel/wsbpel_tut.php [2007, 06/05].

OASIS 2007. Web Services Business Process Execution Language version 2.0. 11 April 2007, OASIS. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

Object Management Group 2006. Business Process Modeling Notation (BPMN) 1.0 Final adopted specification. February 6, 2006, Object Management Group. http://www.bpmn.org/Documents/OMG%20Final%20Adopted%20BPMN%201-0%20Spec%2006-02-01.pdf.

Object Management Group 2005. UML Superstructure, version 2.0. Object Management Group. http://www.omg.org/docs/formal/05-07-04.pdf.

OMG.org 2007. 04/19/2007–last update, Business Process Management (BPMN) Information [Homepage of OMG.org], [Online]. Available: http://www.bpmn.org/ [2007, 05/31].

OMG.org 2004. 10/18/04–last update, Business Process Management Notation (BPMN) Information – FAQ. [Homepage of OMG.org], [Online]. Available: http://www.bpmn.org/Documents/FAQ.htm [2007, 05/31/07].

Ouyang, C., van der Aalst, W.M.P., Dumas, M. & ter Hofstede, A.H.M. 2006. From Business Process Models to Process-oriented Software Systems: The BPMN to BPEL Way, BPMCenter.org. http://is.tm.tue.nl/staff/wvdaalst/BPMcenter/reports/2006/BPM-06-27.pdf.

QPR Software 2007. Interim Report January 1 – March 31, 2007, QPR Software. http://www.qpr.com/www%5CNews.nsf/0/79E33B5E7F3AA852C22572C8002B2BC2/$file/25042007_E.txt.

Recker, J. & Mendling, J. 2006. On the Translation between BPMN and BPEL: Conceptual Mismatch between Process Modeling Languages. In: CAiSE 2006 Workshop Proceedings – Eleventh International Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD 2006), eds. T. Latour & M. Petit. Namur University Press, Luxemburg. P. 521.

Russell, N., ter Hofstede, A.H.M., Edmond, D. & van der Aalst, W.M.P. 2005. Workflow data patterns. Proceedings of 24th Int. Conf. on Conceptual Modeling (ER05). Pp. 353–368.

Smith, H. & Fingar, P. 2003. Workflow is just a Pi process. BPTrends. http://www.bptrends.com/publicationfiles/01-04%20Workflow%20is%20just%20a%20Pi%20Process%20Smith-Fingar.pdf.

van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M., Russell, N., Verbeek, H. & Wohed, P. 2005. Life After BPEL. In: WS-FM 2005, eds. M. Bravetti, L. Kloul & G. Zavattaro. Springer-Verlag, Berlin. Pp. 35–50.

van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B. & Barros, A.P. 2003. Workflow Patterns. Distributed and Parallel Databases, Vol. 14, No. 1, pp. 5–51.

White, S.A. 2004a. Introduction to BPMN. Object Management Group. http://www.bpmn.org/Documents/Introduction%20to%20BPMN.pdf.

White, S.A. 2004b. Process Modeling Notations and Workflow Patterns. Object Management Group.
http://www.bpmn.org/Documents/Notations%20and%20Workflow%20Patterns.pdf.

White, S. 2005. Using BPMN to Model a BPEL Process. BPTrends, Vol. 3, No. 3, pp. 1–18.

White, S.A. 2006. BPMN Tutorial. Object Management Group.
http://www.bpmn.org/Documents/OMG%20BPMN%20Tutorial.pdf.

Workflow Patterns Initiative 2007. n.d.-last update, Workflow Patterns – Standard Evaluations [Homepage of Workflow Patterns Initiative], [Online]. Available:
http://www.workflowpatterns.com/evaluations/standard/index.php [2007, 07/13].

Appendix A: Language comparisons

Table A1. Classification of business process modeling languages.

Language	Visual notation	Specification	Transformation capability
UML 2.0 Activity Diagram (AD)	yes	open	yes
Business Process Definition Metamodel (BPDM)	no	open	yes
Business Process Modeling Notation (BPMN)	yes	open	yes
Event Driven Process Chain (EPC)	yes	proprietary	interchange (academic)[10]
Integrated DEFinition Method 3 (IDEF3)	yes	open	no
Petri Net	yes	open	interchange (academic)[11]
Role Activity Diagram (RAD)	yes	open	no

Table A2. Visual business process modeling language comparison.

Language	Organization	Focus	Portability	Transformability	Status	Support
BPMN 1.0	OMG	executable business processes	poor: XML format not specified	very good: translations covered to a large extent in research and tools; problems result from executable languages	stable; version 1.0 mature, version 2.0 coming	General: very good Transformation: good
UML 2.0 Activity Diagram	OMG	software modeling, business processes	mediocre (XMI format)	mediocre: translations covered partially in research, not much in tools	stable; continuous development	General: very good Transformation: poor

[10] List & Korherr (2006) categorize EPML as an academic proposal for an execution language for EPC, but the referenced publication speaks of ”interchange format”.

[11] Similar to EPC, the interchange format being PNML.

Table A3. Classification of business process model interchange languages.

Language	Format	Executable	Background	Status
Business Process Definition Metamodel (BPDM)	XML	indirectly	industrial	unfinished
Business Process Execution Language (BPEL)	XML	yes	industrial	stable
Business Process Modeling Language (BPML)	XML	yes	industrial	obsolete
Business Process Modeling Notation (BPMN)	visual	indirectly	industrial	stable
Business Process Specification Schema (BPSS)	XML	no	industrial	stable
Event-Driven Process Chain Markup Language (EPML)	XML	no	academic	stable
OWL-Services (OWL-S)	XML	no	academic	stable
The Petri Net Markup Language (PNML)	XML	indirectly	academic	stable
UML 2.0 AD	visual	indirectly	industrial	stable
Web Service Choreography Description Language (WS-CDL)	XML	no	industrial	stable
Web Service Choreography Interface (WSCI)	XML	no	industrial	obsolete
Web Service Choreography Language (WSCL)	XML	no	industrial	obsolete
Web Services Flow Language (WSFL)	XML	yes	industrial	superseded
XLANG	XML	yes	industrial	superseded
XML Process Definition Language (XPDL)	XML	no	industrial	stable

Table A4. Business process execution and interchange language comparison.

Lan-guage	Organiz-ation	Focus	Portability	Transformability	Status	Support/ position
BPEL 1.1/2.0	OASIS	business process orchestration; Web Services	good; some problems between different engines and language versions	good: translations covered to a large extent in research and tools; problems expressing certain diagram constructs	stable	very good: de-facto standard in business process execution
XPDL 2.0	WfMC	business process interchange	very good	very good	stable but fresh	poor; only a few version 2.0 implementations available
BPML	BPMI.org (in the past)	business process execution	unknown	unknown	obsolete	poor; in practice replaced by BPEL
BPDM	OMG	business process interchange	unknown; objective: very good	unknown	unfinished	no tool support; future potential

Series title, number and
report code of publication

VTT Research Notes 2407
VTT-TIED-2407

Author(s)
Koskela, Mika & Haajanen, Jyrki

Title

Business Process Modeling and Execution
Tools and technologies report for SOAMeS project

Abstract

This report presents the results of a survey on business process modeling and execution technologies. The first phase of the research consisted of a broad survey on the available language options. For business process execution, Business Process Execution Language (BPEL for short, officially WS-BPEL or BPEL4WS depending on the version) was considered as the only relevant option. Other executable languages were either obsolete or academic proposals not suitable for industry use. For business process modeling, Business Process Modeling Notation (BPMN) and UML Activity Diagram (AD) were considered suitable. Other available options did not provide enough support for transformations to executable languages.

The expressive power of the languages was evaluated by comparing how well the languages support different workflow patterns. It was found out that there is a significant gap between the expressive power of modeling and execution languages, which means that all models cannot be directly transformed to executable code. Between BPMN and UML AD, the differences in pattern support were minimal. However, it was noted that the specifications are partly ambiguous, which can lead to misinterpretations in the transformations.

The practical utility of the findings was demonstrated by testing two available tools that supported BPMN and BPEL and that were considered prominent based on their documented functionalities. The test results showed that the transformation functionalities were to a large extent dependent on the expressive power of the languages. It was concluded that the technologies have not yet fully matured, but first steps in their adoption can already be taken, because by taking the known shortcomings of the technologies into account in the modeling, automatic transformations from models to code, and even vice versa, can be realized.

ISBN
978-951-38-6958-8 (URL: http://www.vtt.fi/publications/index.jsp)

Series title and ISSN	Project number
VTT Tiedotteita – Research Notes 1455-0865 (URL: http://www.vtt.fi/publications/index.jsp)	6884

Date	Language	Pages
September 2007	English	63 p. + app. 2 p.

Name of project	Commissioned by
Service Oriented Architecture in Multichannel e-Services (SOAMeS)	Tekes - Finnish Funding Agency for Technology and Innovation, VTT, Elisa Oyj, Kesko Oyj, Metsäteho Oy, TietoEnator Processing & Network Oy

Keywords	Publisher
business process modeling, business process execution, business process management, service-oriented architecture	VTT Technical Research Centre of Finland P.O. Box 1000, FI-02044 VTT, Finland Phone internat. +358 20 722 4404 Fax +358 20 722 4374

VTT Tiedotteita – Research Notes

2385 Löfman, Jari, Keto, Vesa & Mészáros, Ferenc. FEFTRATM. Verification. 2007. 103 p. + app. 4 p.

2386 Loikkanen, Torsti, Hyytinen, Kirsi & Koivusalo, Salla. Yhteiskuntavastuu ja kilpailukyky suomalaisyrityksissä. Nykytila ja kehitysnäkymät. 2007. 118 s.

2387 Henttonen, Katja. Stylebase for Eclipse. An open source tool to support the modeling of quality-driven software architecture. 2007. 61 p. + app. 15 p.

2388 Lanne, Marinka & Kupi, Eija. Miten hahmottaa security-alaa? Teoreettinen malli Suomen security-liiketoiminta-alueista. 2007. 52 s. + liitt. 1 s.

2389 Leikas, Jaana & Lehtonen, Lauri. Ikääntyvien idealiike. Käyttäjälähtöisellä innovoinnilla elämänmakuisia mobiilipalveluja. 2007. 34 s.

2390 Tuominen, Anu, Ahlqvist, Toni, Rämä, Pirkko, Rosenberg, Marja & Räsänen, Jukka. Liikennejärjestelmän teknologiapalvelujen vaikutusarvioinnit tulevaisuudessa. 2007. 64 s. + liitt. 5 s.

2391 Mikkola, Markku & Pirttimäki, Antti. Tuotekehitys Kiinassa. Uhka, mahdollisuus vai yhdentekevää? 2007. 31 s.

2392 Kettunen, Jari, Rakshit, Krishanu & Uoti, Mikko. Electronic India. Market trends and industry practices in IT services, telecoms and online media. 2007. 98 p. + app. 2 p.

2394 Herrala, Maila. The value of transport information. 2007. 87 p. + app. 5 p.

2395 Aarnisalo, Kaarina, Heiskanen, Seppo, Jaakkola, Kaarle, Landor, Eva & Raaska, Laura. Traceability of foods and foodborne hazards. 2007. 46 p. + app. 2 p.

2396 Nylund, Nils-Olof, Erkkilä, Kimmo, Clark, Nigel & Rideout, Greg. Evaluation of duty cycles for heavy-duty urban vehicles. Final report of IEA AMF Annex XXIX. 2007. 81 p. + app. 10 p.

2397 Helynen, Satu, Flyktman, Martti, Asikainen, Antti & Laitila, Juha. Metsätalouteen ja metsäteollisuuteen perustuvan energialiiketoiminnan mahdollisuudet. 2007. 66 s.

2398 Jansson, Kim, Mikkola, Markku & Ryynänen, Tapani. Verkostoyhteistyöllä Kiinaan? SeaChi-projektin loppuraportti. 2007. 46 s. + liitt. 6 s.

2399 Hänninen Hannu, Brederholm, Anssi, Saukkonen, Tapio, Gripenberg, Hans, Toivonen, Aki, Ehrnstén, Ulla & Aaltonen, Pertti. Hot cracking and environment-assisted cracking susceptibility of dissimilar metal welds. 2007. 182 p.

2400 Ailisto, Heikki, Matinmikko, Tapio, Häikiö, Juha, Ylisaukko-oja, Arto, Strömmer, Esko, Hillukkala, Mika, Wallin, Arto, Siira, Erkki, Pöyry, Aki, Törmänen, Vili, Huomo, Tua, Tuikka, Tuomo, Leskinen, Sonja & Salonen, Jarno. Physical browsing with NFC technology. 2007. 70 p.

2401 Häkkinen, Tarja, Vares, Sirje, Huovila, Pekka, Vesikari, Erkki, Porkka, Janne, Nilsson, Lars-Olof, Togerö, Åse, Jonsson, Carl, Suber, Katarina, Andersson, Ronny, Larsson, Robert & Nuorkivi, Isto. ICT for whole life optimisation of residential buildings. 2007. 207 p.

2403 Toivonen, Santtu. Web on the Move. Landscapes of Mobile Social Media. 2007. 56 p. + app. 3 p.

2404 Vares, Sirje & Lehtinen, Jarkko. Lasipakkausten keräysjärjestelmän tehostaminen ja lasin hyötykäytön ympäristövaikutukset. 2007. 122 s.

2407 Koskela, Mika & Haajanen, Jyrki. Business Process Modeling and Execution. Tools and technologies report for SOAMeS project. 2007. 63 p. + app. 2 p.

VTT
PL 1000
02044 VTT
Puh. 020 722 4404
Faksi 020 722 4374

VTT
PB 1000
02044 VTT
Tel. 020 722 4404
Fax 020 722 4374

VTT
P.O. Box 1000
FI-02044 VTT, Finland
Phone internat. + 358 20 722 4404
Fax + 358 20 722 4374

ISBN 978-951-38-6958-8 (URL: http://www.vtt.fi/publications/index.jsp)
ISSN 1455-0865 (URL: http://www.vtt.fi/publications/index.jsp)