

Project Number: **215219**
 Project Acronym: **SOA4All**
 Project Title: **Service Oriented Architectures for All**
 Instrument: **Integrated Project**
 Thematic Priority: **Information and Communication Technologies**

D6.3.1. First Specification of Lightweight Process Modelling Language

Activity:	Activity 2 - Core R&D Activities	
Work Package:	WP 6 - Service Construction	
Due Date:	M12	
Submission Date:	09/03/2009	
Start Date of Project:	01/03/2008	
Duration of Project:	36 Months	
Organisation Responsible of Deliverable:	SAP	
Revision:	1.2	
Author(s):	Florian Schnabel	SAP
	Matthias Born	SAP
	Lai Xu	SAP
	Rafael González-Cabero	ATOS
	Freddy Lecue	UNIMAN
	Nikolay Mehandjiev	UNIMAN
	Tomas Pariente (Reviewer)	ATOS

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	X

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	02/12/08	ToC defined	Florian Schnabel
0.2	13/01/09	Section 7 Content	Freddy Lecue, Nikolay Mehandjiev (UniMan)
0.3	16/01/09	Section 3, 4, and 5	Lai Xu (SAP)
0.4	19/01/09	Section 2, Section 6	Sven Abels (TIE), Rafael González-Cabero (ATOS), Adrian Mos (INRIA)
0.5	27/01/09	Section 7	Freddy Lecue (UniMan)
0.6	30/01/09	Section 3, 4, and 5	Born, Matthias (SAP), Florian Schnabel (SAP), Lai Xu (SAP)
0.7	05/02/2009	Revise	Lai Xu (SAP)
0.8	05/02/2009	Section 4.3	Rafael González-Cabero (ATOS)
0.9	05/02/2009	Revise Structure	Lai Xu (SAP)
1.0	06/02/2009	New figures and general revision	Rafael González-Cabero (ATOS)
	19/02/2009	Internal Review	Jean-Pierre LORRE (EBM)
	19/02/2009	Internal Review	Tomás Pariente Lobo (ATOS)
1.1	25/02/2009	Revise	Lai Xu (SAP)
	27/02/2009	Update Section 6.1	Freddy Lecue (UniMan)
	27/02/2009	Section 3.1 and Revise	Lai Xu (SAP)
1.3	04/03/2009	Update Section 5.1	Rafael González-Cabero (ATOS)
	05/03/2009	Review	Gianluca Ripa (CEFRIEL)

1.4	05/03/2009	Revise	Lai Xu (SAP)
Final	09/03/2009	Overall format and quality revision	Malena Donato (ATOS)

Table of Contents

EXECUTIVE SUMMARY	8
1. INTRODUCTION	9
1.1 PURPOSE AND SCOPE OF THE DELIVERABLE	9
1.2 STRUCTURE OF THE DOCUMENT	10
2. SUMMARY OF THE REQUIREMENTS FOR A LIGHTWEIGHT, CONTEXT-AWARE PROCESS MODELLING LANGUAGE	11
2.1 METHODOLOGY FOR REQUIREMENTS ACQUISITION	11
2.2 SOA4ALL GENERAL REQUIREMENTS	12
2.3 REQUIREMENTS FROM THE USE CASES	12
2.3.1 <i>WP7 Requirements: End-user Integrated Enterprise Service Delivery Platform</i>	12
2.3.2 <i>WP8 Requirements: W21C BT Infrastructure</i>	13
2.3.3 <i>WP9 Requirements: C2C Service eCommerce</i>	14
2.4 SUMMARY OF REQUIREMENTS	15
3. DESIGN PRINCIPLE OF THE LIGHTWEIGHT, CONTEXT-AWARE MODELLING LANGUAGE	17
3.1 DIFFERENT ABSTRACTION LAYERS FOR LIGHTWEIGHT PROCESS MODELLING LANGUAGE	18
3.2 REFLECTION ON REQUIREMENTS	19
4. CONTEXT-AWARE PROCESS MODELLING	21
4.1 METHODOLOGY	22
4.2 CONTEXT-DRIVER PRINCIPLE	23
4.3 META-MODEL	25
4.3.1 <i>Business Entities</i>	27
5. LANGUAGE SYMBOLS, PROCESS PATTERNS, AND WORKFLOW TEMPLATES	29
5.1 SYMBOLS USED IN LIGHTWEIGHT PROCESS MODELLING LANGUAGE	29
5.2 DESCRIPTION OF PROCESS PATTERNS	30
5.3 WORKFLOW TEMPLATES	34
5.4 GOALS AND TEMPLATE PROCESSES	36
5.5 META-MODEL OF THE LIGHTWEIGHT PROCESS MODELLING LANGUAGE	39
5.6 SUMMARY AND REMARKS	40
6. LANGUAGE EVALUATION	42
6.1 LAB EXPERIMENTS	42
6.2 FEEDBACK FROM USE CASES	42
7. CONCLUSIONS	44
8. REFERENCES	45
ANNEX A. PROCESS PATTERNS AND TEMPLATES IN YAWL	47

Glossary of Acronyms

Acronym	Definition
B2B	Business-to-Business
BPA	Business Process Analysis
BPEL	Business Process Executable Language
BPM	Business Process Management
BPML	Business Process Modelling Language
BT	British Telecom
CCTS	Core Components Technical Specification
D	Deliverable
ebXML	Electronic Business using eXtensible Markup Language
EPC	Event-driven Process Chains
IDE	Integrated Development Environment
IT	Information Technology
PHP	PHP Hypertext Pre-processor
QoS	Quality of Service
SAP	Systeme Anwendungen und Produkte
SDK	Software Development Kit
SOA	Service-Oriented Architecture
T	Task
UML	Unified Modelling Language
WP	Work Package
WSMO	Web Service Modelling Ontology
YAWL	Yet Another Workflow Language

List of Figures

Figure 1 Example of the Invoice Processing Process at Business Level.....	21
Figure 2 Context-aware Business Process Model	23
Figure 3 Simplified Meta Model of the Context Driver Principle.....	24
Figure 4 Meta Model.....	26
Figure 5 Sequence in BPMN.	30
Figure 6 Parallel Split in BPMN.....	31
Figure 7 Synchronization in BPMN.	31
Figure 8 Exclusive Choice in BPMN.	32
Figure 9 Simple Merge in BPMN.	32
Figure 10 Multi-choice in BPMN	33
Figure 11 Synchronizing Merge in BPMN.....	33
Figure 12 Multi-Merge in BPMN.....	33
Figure 13 Discriminator in BPMN.....	34
Figure 14 Task pay is executed each time one of the three preceding task completes	35
Figure 15 Task pay is executed only once, i.e., after all started tasks have completed.....	35
Figure 16 Task pay is executed only once, i.e., after the first task has completed.	35
Figure 17 Composite Goals and Process Templates Graphical Representation.	37
Figure 18 Atomic Goal Graphical Representation.....	37
Figure 19 Goal Main Components.	38
Figure 20 Adapter main components.....	38
Figure 21 Meta Model of the Lightweight Process Modelling Language.....	40
Figure 22 Sequence in YAWL.	48
Figure 23 Parallel Split in YAWL.....	49
Figure 24 Synchronization in YAWL.	49
Figure 25 Exclusive Choice in YAWL.....	50
Figure 26 Simple Merge in YAWL.....	50
Figure 27 Multi-choice in YAWL.....	51
Figure 28 Synchronizing Merge in YAWL.	51
Figure 29 Multi-Merge in YAWL.....	52
Figure 30 Discriminator in YAWL.....	52
Figure 31 Task pay is executed each time one of the three preceding task completes.	53

Figure 32 Task pay is executed only once, i.e., after all started tasks have completed.	53
Figure 33 Task pay is executed only once, i.e., after the first task has completed.	54

List of Tables

Table 1 Symbols used in Lightweight Process Modelling Language.	29
Table 2 Feedback from WP8.	42
Table 3 Symbols used in YAWL.	48

Executive Summary

SOA4All concentrates on bringing IT solutions to non-technical users. Existing process modelling languages and special executable process modelling languages are not designed for non-experienced users [29]. Lightweight process modelling seeks to lower the barrier to entry for process modelling. Non-experienced users get advanced guidance during the modelling activities. Lightweight process modelling supports modelling in different abstraction levels and allows switching or drilling between those levels, e.g., coming from a high-level process to a more detailed sub process.

In the context of lightweight process modelling, a new and easy to manage process modelling language will be required, since existing languages are too complex for non-technical users. Besides the requirement to manage easily the process-based service composition models, another requirement for the modelling language is to allow for the specification of templates. The definition of such a modelling language is the main goal of task 6.3, a first specification draft is provided by this deliverable. If such a language is to be usable by people who are not software professionals trained in management of complexity and abstraction, it should hide as much of the service composition complexity as possible. Nevertheless, we should provide sufficient notational semantics for users to understand the unavoidable interactions between services being composed; and sufficient expressive power for the users to construct useful compositions.

Another aspect of the lightweight modelling language is to allow for the definition of contextualized processes by supporting the specification of context information sources and their role they play in the process. Hence it will leverage the ontologies defined in WP3 for modelling the context.

Finally, this language should allow for the definition of domain-specific building blocks that can be reused by other users.

In this document, ***D6.3.1. Specification of Lightweight, Context-aware Process Modelling Language***, we present a first design of the lightweight process modelling language. The document is based on the state-of-the-art report contained in deliverable D6.1.1, use case requirements, conceptual considerations, and usability studies. The report contains the specification of the first set of design elements. Further, this deliverable contains a first notion on the evaluation of the modelling language.

1. Introduction

1.1 Purpose and Scope of the Deliverable

Why do we need a lightweight process modelling language?

At the moment, existing languages for business process modelling address different purposes. The more formalized the models are, the less ambiguities exist in interpreting these models. However to create highly formalized process models requires high modelling skills.

Lightweight process modelling seeks to lower the barrier to entry for process modelling. Non-experienced users get advanced guidance during the modelling activities. Errors, misspellings and inconsistencies should be avoided from the beginning. Lightweight process modelling supports modelling in different abstraction levels and allow switching or drill-down between those levels, e.g. coming from a high-level process to a more detailed sub process.

Existing process modelling languages and special executable process modelling languages are not designed for non-experienced users. To summarize there is a clear need to support the non-trained user in creating formalized process models that either can be easily transformed into an executable modelling language or be directly executed. The envisioned lightweight process modelling language will allow for the easy creation of formalized process documentation models as well as allow for the creation of executable process models out of the specification on a high abstraction level.

Who will use the lightweight process modelling language?

The business cases described late in this document make clear statements towards the necessity of supporting the end user rather than the programmer. Hence, it is important to define who actually is the end user. There is a broad variety of user groups ranging from management to business analysts who are possible stakeholders in a process management project. A framework for the classification of users has to be built. Some criteria for differentiating users can be the modelling purpose, their business and IT skills and their capability to abstract from specific application scenarios.

How will the target model or target application be used?

In addition, the specific target model or target application needs to be carved out more clearly. Is it built for automating processes, collaboration between organisational units or is it consultation and documentation? We assume that in most application scenarios it will be a mix. Another challenge is potential collaboration in defining processes. We can think about a manager approving a process model. Based on these premises a methodological framework for the access and creation of models needs to be developed.

How SOA4All will contribute to these challenges?

SOA4All concentrates on bringing IT solutions to non-technical users. In the context of lightweight process modelling a new, easy to manage modelling language will be required since existing languages are too complex for non-technical users. Besides the requirement to manage easily the process-based service composition models another requirement for the modelling language is to allow for the specification of templates. A first draft of such a modelling language is the main goal of this deliverable. If such a language is to be usable by people who are not software professionals trained in management of complexity and abstraction, it should hide as much of the service composition complexity as possible. It should also provide sufficient notational semantics for users to understand the unavoidable

interactions between services being composed; and sufficient expressive power for the users to construct useful compositions.

Another aspect of the lightweight modelling language is to allow for the definition of contextualized processes by supporting the specification of context information sources and their role they play in the process. Hence, it will leverage the ontologies defined in WP3 for modelling the context.

Further, we will allow for the definition of domain-specific building blocks that can be reused by other users. An example of such building blocks can be found in the EU FP6 PICTURE project [26].

Lastly, we have to define the generation of appropriate artefacts from the modelling language constructs. In particular, we need to generate the executable process descriptions as well as the components and entities needed by the runtime infrastructure (such as distributed service bus descriptions, and architectural composite descriptions). Furthermore, information from the language elements will be used in the generation of monitoring / provenance information (such as which process descriptions a particular service is in). The generation of the executable process descriptions will make use of context information. Hence we will use an existing framework for accessing the context in which a service is deployed. This will enable service adaptation as addressed by the Tasks 6.4 and 6.5 in SOA4All.

The process editor for the lightweight process modelling language that makes the functionalities defined above available to SOA4All users will be developed in Task 2.6 and integrated into the SOA4All Studio.

1.2 Structure of the Document

This deliverable is based on the state-of-art report and requirements for service construction D6.1.1 [29]. It is organised as follows Section 1 gives an introduction to the scope and content of this deliverable. The requirements for the lightweight process modelling language are covered by Section 2. These requirements result out of the SOA4All use cases as well as from general user needs. In Section 3, design principles of lightweight, context-aware modelling language are introduced. Context-aware process modelling is discussed in Section 4. Language symbols, process patterns, and process templates are presented in Section 5. Section 6 includes the concept for the evaluation of the process modelling language and some feedback from WP8 and WP9 on process patterns and workflow patterns. Finally, the conclusion section summarizes the deliverable and provides an outlook on future steps in T6.3.

2. Summary of the Requirements for a Lightweight, Context-aware Process Modelling Language

After studying use case documents from WP7, WP8 and WP9, i.e. D7.1, D7.2, D8.1, D8.2 and D9.1, we must conclude the documented case studies are written on a very high abstraction level. It can be well understood that as SOA4All is an advanced research project, it is not easy to acquire requirements from real life. It requires more sophisticated skills to achieve our goals of collecting reasonable requirements.

One of the important requirements for process modelling in SOA4All is the need to be lightweight. Lightweight, however, is a relative concept. What do we compare with? Lightweight in which dimension? Should the produced lightweight models be concise? Should the language have only a few symbols? Should the language be very easy to use for beginners? Should there be few requirements on systems that can execute the models? Etc.

One of the criteria of being lightweight is end-user friendliness. This brings another problem. Who are actually our end users? The end users are civil servants, upstream and downstream customers, entrepreneurs, and so on. But these categorisations do not provide any interesting information on their business or IT skills. Besides, EPC (Event-driven Process Chain), UML and Petri Nets have been popular in industry for long time. BPMN, BPEL and YAWL are also well accepted by certain groups of people. There is another factor. Microsoft office assistant (or clippie) has been developed as a user-friendly tool. But it is pretty annoying for most people. This shows that user friendliness is not easy.

For questions like who our end-users are and what kinds of business and IT/technical skills we can expect from them, we have initiated a discussion with the authors of the use case deliverables. Although a definite answer of who are end users and what kinds of skill they have cannot be provided, we do believe that they may not be able to create a complex process model. They should however be able to read a reasonably simple process model. Therefore, the *end users* are not typically paid to do the job of a professional modeller or programmer, and any programming or modelling efforts that they perform tend to be basic and only applied to the extent that they solve the business problem at hand.

In the following sections, we present our methodology for requirements acquisition. We list SOA4All general requirements and requirements from the user cases respectively. Finally, we summarize the requirements for designing the lightweight, context-aware process modelling language.

2.1 Methodology for Requirements Acquisition

A *pattern* is an abstraction from a concrete form that keeps recurring in specific non-arbitrary context [15]. The use of patterns is a proven practice in the context of object-oriented design, as evidenced by the impact made by the design patterns of Gamma et al. (1995)[14].

We thus think about providing some process modelling patterns, modelling fragments/process modelling templates and even some completed process models of typical cases to our end users. This assumes that the end users can read process models. Similar to programming languages, there are different process modelling languages available, but familiarity with one of them makes it easy to understand others. By comparing, and more specific by pointing out the differences of similar models, the end users edit existing templates and run their processes. This way may get the end users on their ways of modelling processes easier.

Our colleagues in WP7, WP8 and WP9 are working hard to acquire requirements and to

refine the sequence of activities into detailed executable processes. Looking at the process patterns we provided, it can hopefully help our colleagues to get deeper details out of the current cases. Requirement acquisition is an incremental process. They could provide us the feedback about what process patterns and templates are well used in different domains. We would like to follow the agile method [29] to fast providing our process patterns and templates to WP7, WP8, and WP9. After receiving feedback from them and detailed requirements of the lightweight process modelling languages, we will enhance the process patterns and templates, and justify our design of the language. Finally, we will identify our well-used process patterns and templates getting in the end, and the lightweight process modelling language can be well accepted by the users.

2.2 SOA4All General Requirements

One of the objectives of SOA4All is to open the world of service composition to the non-technical user. Depending on the user's skills and knowledge, we should allow the user to model its services and processes on different levels of complexity. Hence, SOA4All seeks to integrate all kinds of experts and non-experts. Graphical modelling elements will facilitate the process composition.

The process models will be used to create representations meeting both the business and technical needs of users and their use cases. Achieving the objectives of SOA4All requires the delivery of a service and process composition interface, the SOA4All process editor, considering the skills and tasks of our target users. The long-term aim of SOA4All is to open up process modelling to everyone, yet at first instance our target users are those found in the SOA4All case studies (WP7-9).

The case studies are still in the stage of initial definition, yet the following characteristics of our end users are clear:

- Most target users will have professional background
- Some target users will not be professional software developers and would not have received significant training in programming nor system design
- Most target users will be experts in the tasks and processes they are trying to support by using SOA4All

Therefore in the context of SOA4All, we need to develop an intuitive process modelling tool, which is able to use by someone without too much prior instructions. A certain level of technical competence and familiarity with process models will have to be assumed.

2.3 Requirements from the Use Cases

2.3.1 WP7 Requirements: End-user Integrated Enterprise Service Delivery Platform

The goal of WP7 is build an integrated demonstrator, which integrates SAP enterprise services into the SOA4All platform. This demonstrator will allow civil servants to handle typical administrative procedures (such as a permit approval process). More specifically, using the web-based tools of the SOA4All Studio, public servants can search, model, annotate, modify, share, analyze, and execute administrative procedures in the form of lightweight business processes. These processes may be composed of enterprise services (hosted by SAP), public web services (hosted by third party service providers), and human activities (to be executed by end users). For public administrations, the main benefit of such a flexible and open service delivery platform is the possibility to quickly address new

challenges and requirements, e.g., such as the ones formulated by the EU Services Directive.

Let us now enumerate the main requirements that we have identified as relevant to the work package

- The models and tools should support a range of different users with different roles and skills, in the concrete context of this refers to
 - Front office: high-level knowledge of all processes
 - Back-office: very detailed knowledge of selected processes
- Processes should have associated meta-data properties that allow for the process retrieval using properties such as name, author, category, data objects, user comments, etc.
- Processes should be reusable. A process or parts of a process should thus be more like building blocks that can be recombined into more complex ones.
- The process model should provide a graphical representation of processes. It will make them easier to create and easier to share
- The process steps should enable the representation of:
 - A selection of a concrete service acting as a service instance
 - A service class containing a set of similar services
 - Services templates, similar to service classes, but with some information left intentionally unspecified.
 - Goals, describing functional properties of services as well as preconditions and effects
- Various sorts of filling parameter descriptions of process steps and processes as a whole should be allowed
 - Dynamic input parameters that could be filled out by output parameters of preceding services or context-dependent parameters.
 - Static input parameters that always have the same value
 - Input parameters provided by the user (via browser-based UI) or by automatic information sources (services output or current context).
- Processes should be described with enough abstraction and freedom to allow:
 - transparent deployment on demand
 - multiple and parallel running instances per process

The above requirements are based upon deliverables D7.1 and D7.2. However, as the delivery of the latest version of D7.3 has been postponed until M13, this deliverable can not reflect the final requirements of WP7.

2.3.2 WP8 Requirements: W21C BT Infrastructure

Web21C is the name currently given to the platform over which BT will provide next generation services on top of its all IP-based 21st Century Network (BT 21CN). Some of

these services will be provided by BT and others will be provided by third parties. Web21C is central to BT's transformation from a traditional telecommunications company to a converged software and services business. Web21C will allow third parties to use BT's network as a platform for delivery of their services, for which BT get revenue. These are not typically other network competitors, but a new breed of partner - software companies, developers and content providers.

Currently Web21C comprises of a set of Web services, and software development kits (SDKs) that provide external access to a number of BT capabilities, such as making a voice call and sending an SMS text message.

In the following, we will identify different requirements from each of the scenarios that have been defined for this use case. From the **Web21c Telco application design** scenario (casual-user side) we identify the following:

- The representation, tools and techniques that will be developed to compose services should envisage that different communities might generate compositions, which can be either internal or external to the telecommunication company.
- Services compositions should be based on different criteria, namely functionally based, non-functional based (e.g. QoS), user goal based, and context-based.
- The lightweight process model (and the overall service construction environment) should be easy to use, lowering thus the entry barrier in using the composition. That includes:
 - Users do not need to have any programming experience (e.g. in using an IDE to program in Java, C# or PHP)
- The semantic descriptions both of services and the process as a whole should be formally defined in order to automate tasks such as suggesting compatible services in service compositions.

From the **Business Reseller scenario**, we identify the following requirement

- The lightweight process model should contain information about the QoS, context criteria, etc. This information can be used later on for ranking processes, monitoring, logging etc.

2.3.3 WP9 Requirements: C2C Service eCommerce

WP9 *C2C Service eCommerce* use case will be entirely focused on providing an easy way for end users to use third party services offered through the framework. In this use case the SOA4All platform should enable users to build eCommerce applications in order to market and sell their own products, such as photos or furniture or by providing their own innovative services built from a mash-up of existing service offers. End customers are able to use various SOA4All-enhanced tools offered through this framework to build their own end customer applications. While people may use the SOA4All results to build generic applications, the eCommerce framework will provide eCommerce specific functionality and will itself also use the SOA4All services for achieving this. For example, it will provide typical eShop functionalities such as a shopping cart feature and an access to payment providers using the SOA4All service orchestration and communication facilities. More precisely, WP9 will provide services for different eCommerce areas such as advertisement, marketing, distribution, and payment, based on existing partner products and services. In addition, the inclusion of additional third party services via a service broker will be enabled.

The requirements for this work package of this deliverable are:

- The end users can build their eShop using modules such as product management, categories, shopping cart, stock, payment and delivery options and services, etc.
- The typical users of the WP9 need a simple way to describe their requirements without knowing of different versions or split and join types of a process model and how tokens are processed in the process model.
- The users of the eCommerce application would create suitable compositions of services, based on workflow templates, which replaces the usual order process for a customer of the eCommerce application. Design of workflow templates need to be a simple formalism, which has the ability to exchange some of the activities with real services.
- A service broker combines services to provide “service bundles” to the end users. These services bundles consist of simple processes, for example, the combination of fraud detection and address check services with the actual payment service.

2.4 Summary of Requirements

Now that we have discussed the most important requirements of the lightweight, context-aware process modelling language, based on the use cases from WP 7, WP8, and WP9. It would be useful to summarize them:

- **Executable process modelling language:** It does not matter how user friendly it is, at a certain level, the different logical splits and merges are still needed. In other words, although simple for end users, the language should be expressive and well-defined enough in order to be executable.
- **Easy to use for end users:** The lightweight, context-aware process modelling language will be used by users without professional modelling skills. The complexities of process creations need to be hidden somehow.
- **Integrated with end user’s daily work life:** In order to reap the full potential of SOA4ALL platform, lightweight process modelling needs to be integrated in the way people do their work, i.e. directly within the context of their work.
- **Annotated and well-managed process templates:** Process templates can be pre-defined, or created by users and validated by experts. All templates can be classified to share with all or within small communities, and serve as flexible basis for the definition of processes.
- **Specified constraints:** It must be possible to specify some constraints such as a constraint that certain tasks or activities in the process model must be performed by certain pre-required Web services.
- **Fault-handling:** Reporting modelling errors and potential during the process creation stage.
- **Non-functional description of tasks in a process:** We did not receive a concrete requirement in regard to non-functional aspects of tasks in a process yet. It can be specified in the process modelling stage. It can also be done in the process execution stage, i.e. invoking Web services. We are still discussing this with our partners.

In this section, we have reviewed requirements from general aspects and the SOA4All use cases. In the next section, we will present our design principle of the lightweight, context-

aware modelling language. It is worth noting however, that at the time of this writing, the use cases from WP7, WP8 and WP9 are in an early stage and it is therefore not possible to provide a complete and thorough alignment.

3. Design Principle of the Lightweight, Context-aware Modelling Language

In the previous section, we have reviewed the requirements from a general perspective and from the perspective of the SOA4All use cases. In this section, we look at current technology support for lightweight process modelling. We further present our design principle of the lightweight, context-aware modelling language.

Lightweight process modelling is a combination of techniques that seek to lower the entry barrier for process modelling. This includes fostering a more participative style of modelling and providing a forum for the community of experts. The following technical requirements arise:

- **Easy access:** Business user enablement demands a focus on usability. This comprises both simplicity of the approach and a low footprint of the solution (“zero install”). Ideally, users can model processes using simplified notations in a Web 2.0 environment and draw on modelling best practices.
- **Process Wiki:** Combine both structured and unstructured information and publish them as a single point of reference to the organisation. The Process Wiki invites the community to participate in discussions, and to provide comments or ratings in relation to process proposals. Process documents can be generated for offline reading and dissemination.

The lightweight process modelling tool as being developed in WP2 has to provide an intuitive user interface. Furthermore, it has to provide advanced guidance during the modelling activities. Errors, misspelling and inconsistencies should be avoided from the beginning. The integration of different views helps to provide only relevant information to a certain user group. The tool should support modelling in different process abstraction levels and allow switching or drill-down between those levels, e.g. coming from a high-level process to a more detailed sub process. The solution has to provide process governance. A process owner needs to be assigned to every process. The process owner is responsible for answering questions regarding the model, monitoring activities related to it, and keeping the model up-to-date. Another important aspect is to provide collaborative modelling functionalities. Process models should be easily accessible and understandable for all participants.

The lightweight process modelling tool should provide a proper process content repository which offers easy access to all artefacts of a business process in order to store, update, retrieve, and delete information relevant to the process. The process repository should store process artefacts with the same semantic meaning only once and in a structured manner to avoid redundancy. In addition, the repository has to be extensible with predefined interfaces to allow customized changes. The modelling tool should provide a pre-defined knowledge base containing reference models, best-practise models and further examples.

In order to simplify business process modelling, models must be highly reusable, favouring process flexibility and minimizing designs made from scratch. There is wide agreement that patterns can accelerate the process of designing a solution and reduce modelling time. Patterns enable participants of a community to communicate more effectively, with greater conciseness and less ambiguity [22], [23], [24]. We thus choose to use a part of well-known workflow patterns from [12] as our *process pattern*. The patterns range from very simple to very complex and cover the behaviours that can be captured within most business process models. Workflow patterns also have a well-defined formal foundation. It will provide special value when we introduce context-awareness into the language. The context-awareness

principle of the process modelling will be further discussed in Section 4.

Besides, applying process patterns at the process modelling stage, end users can obtain support in case the application of a pattern causes a modelling error and the sequence of applied patterns can be traced during process editing time.

The workflow patterns from [12] are however too fine-grained and not sufficiently enriched with information on the context and consequences to represent a reusable solution. Therefore, we introduce *workflow templates* that are different combinations of process patterns. The processes represented by a workflow template are sound. Certain workflow templates can be enriched with the information that they are valid for different domains, i.e. business context.

Where process patterns provide flexibility and guidance during the design phase, there is another opportunity for process flexibility. Process activities are traditionally concrete and bound to services or other means of implementation at design time. We use activity goals as unbound activities that are bound to a particular service at runtime. Activity goals specify the conditions for their implementation in such detail, that it is possible to automatically find fitting services with a high degree of accuracy.

In short, our design principle of the lightweight, context-aware modelling language can be summarized as:

- Context-awareness

The names of activities/tasks involved in a process model should be unified. The context-driver principle allows identification, storage, and representation of a business process artefact only once. A business process is instantiated depending on specific context categories (e.g., business process, industry, country, business role, etc.). Providing context information such as “country” information, a specific model can be invoked out of the same name process models during the modelling stage. For example, in different countries there is a different banking system that allows different types of payment. More details can be found from Section 4.

- Usability and Reusability

We adopt seven symbols from BPMN and add two goal related notations for describing a control flow of a process now. However, we do support different process patterns in our language. Details can be found in Section 5.1 and 5.2. Process patterns, workflow templates, process fragments, reference models, best-practice models, and further example models are provided in association with the language. Details can be found from Section 5.3.

- Flexibility

Unlike developing an activity in a traditional workflow system, implementations of activities/tasks are fixed. We use activity goals as unbound activities. Activities/tasks can thus invoke different Web services. Description of an activity goal can be found from Section 5.4.

3.1 Different Abstraction Layers for Lightweight Process Modelling Language

In order to keep a balance between simplicity of use and expressive power of the lightweight process modelling language, we have different abstraction layers for different purposes. For

end users, the T2.6 process composer provides four symbols (see D2.6.1 Section 6.1). Gateways have been explicitly omitted as they are believed to be hard to understand for non-technical users [29]. They are implicitly modelled by multiple outgoing and incoming connections. More details can be found from D.2.6.1.

This deliverable introduces an executable process language for lightweight process modelling and advocates our context-aware process modelling method. We provide limited graphical notations to express reasonable complex control flows. Rich process patterns and workflow templates are provided to support “modelling-by-example”.

For making the graphical process modelling language available for execution, we have considered XPDL [27] or WSBPEL [28] for a representation of the language that is supported by process execution engines. A future selection of a representation of the executable process modelling language will also be based on the semantic matching of those languages with our language. A comparison of XPDL and WSBPEL can be found in [31]. The final choice will also depend upon the chosen execution engine within the project. We need to further extend the selected XML-based language representation by semantic annotations and other language properties.

3.2 Reflection on Requirements

We reflect upon our design of a lightweight process modelling language to requirements we summarized in Section 2.4.

For the “executable process modelling language”, the balance between simplicity of use for end users and expressive power of the language is a key aspect. We keep logic expression of a control flow as simple as possible, but it needs to be sufficient for execution. It is possible for a process editor (T2.6) to support a subset of the language and to allow users to create their models in more flexible ways.

For “ease of use for end users”, annotated workflow templates can be easily discovered by end users. For some end users, their requirements can already be satisfied by choosing a good workflow template and running it. For other end users who want to create a new process by editing an existing workflow template or by integrating different workflow templates, we will provide context-aware guidance during the process modelling stage. Learning and training effort should be minimized. Different workflow templates and models can also be ranked based on popularity in a community-centric SOA4All environment.

For “integration with end user’s daily work life”, we provide context aware business vocabularies which extend technical vocabulary and can be used for the process modelling in order to keep language consistent across the different workflow templates, process fragments and process models.

For “annotated and well managed workflow templates”, a user-friendly annotation of semantic description of workflow templates should be provided. Semantic technologies are further suited to describe workflow templates and their interdependencies and relationships. It should allow end users to easily discover differences between similar workflow templates. SOA4All establishes a dynamic platform where workflow templates, process fragments, and process models are contributed, grouped, consumed, and managed.

For “specified constraints”, before process execution, end users are able to specify that certain tasks or activities in the process model must be performed by certain pre-required Web services. It means that the selection of other Web services for executing the process is

based on the pre-required Web services.

For “fault-handling” during the process modelling stage, we apply some general design rules to guarantee soundness of a process model, i.e. checking deadlocks, lack of synchronization, and so on.

4. Context-aware Process Modelling

Current process modelling approaches and tools are still very costly and error-prone since users are not guided in any sensible way. The analysis of Gartner shows that BPA (Business Process Analysis) tools are too complex for the average user and that the major modelling tool with a market share of 30% is MS Visio [1]. Although a variety of BPM tools and different BPM methodologies is available, none of the tools seems to appropriately address the modelling needs. Vendors still construct their own model representations and tools, which are mostly not interchangeable, and yet most business analysts are using office tools (like Microsoft PowerPoint or Microsoft Excel) to describe their process landscape. Instead of reusing existing process models or parts of process models (also called process artefacts in the following), they are usually replicated and modelled from scratch.

Process models are processes of the same nature that are classified together into a model. Thus, a process model is a description of a process at the type level. Since the process model is at the type level, a process is an instantiation of it. The same process model is used repeatedly for the development of many applications and thus, has many instantiations. One possible use of a process model is to prescribe how things must/should/could be done in contrast to the process itself which is really what happens. A process model is roughly an anticipation of what the process will look like. What the process shall be will be determined during actual system development [3]. An example of a process model for invoice processing is depicted in Figure 1.

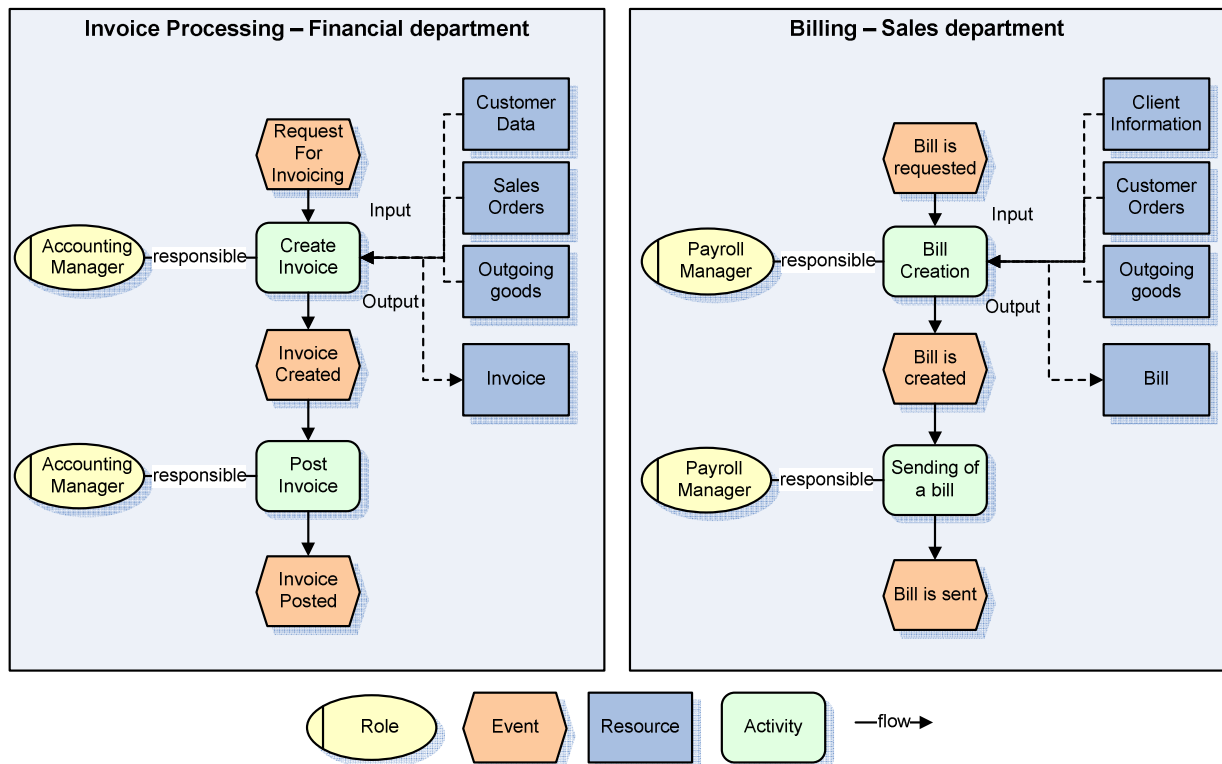


Figure 1 Example of the Invoice Processing Process at Business Level

Let us assume that there exist two departments, namely the financial department and the sales department within a company, which are using their own terminology in order to model and describe the processing of an invoice (compare Figure 1). As the naming of business process artefacts is often more art than science [2] the modellers often do not apply any naming conventions and name business artefacts in arbitrary ways (compare Figure 1). Furthermore, Figure 1 shows that both processes actually describe the same process, while both departments use their own terminology.

We can distinguish two basic levels of reusability: Reusability of the semantic representation and of the structural characteristics of process artefacts. In the following, we argue that reusability on both levels is hardly given.

- I. **Reusability across process models**, i.e., parts of a process model, e.g., the activity “Create Invoice” in the Invoice processing process model in Figure 1, typically also occur in other process models (sales, purchasing). However, creators of process models users typically come up with different labels for the same activity (e.g., another user might call the activity “Creation of an invoice”).
- II. **Reusability of structural dependencies**, i.e., elements of the process model are typically linked to one or more elements. As an example, let us come back to the process step “Create Invoice” in our running example. This process step is linked to the responsible role “Accounting Manager”. However, in a different process model the process step might be linked to a responsible role “Solution Manager”.

The examples already indicate that both types of reusability are problematic. We propose to integrate context-awareness into business process modelling and thus allow a higher flexibility and reusability of process artefacts. A process model is defined by its process artefact and the context in which it is used. The general idea is to create a standardized, consistent, and understandable description of every business process artefact using ontologies and to link each artefact to a specific business context. The primary purpose is to achieve consistency in the naming and to facilitate the understandability of the business process models depending on a business context. As a result, the usage of insufficient business artefacts and terminology can be avoided. Examples of insufficient terminology are words which have the same spelling and pronunciation but have different meanings (homonyms), or words that have the same spelling but different pronunciation and different meanings (heteronyms).

4.1 Methodology

In order to overcome the issues of reusability and misleading terminology we propose to introduce context-awareness into business process modelling. The general idea is that every business artefact is assigned to a business environment called business context. A *business context* describes where this business artefact is valid. For example, certain workflow templates can be enriched with the information that they are only valid within a certain business domain i.e. business context.

Before a modeller is now able to create process models, first he has to define in which business context he is modelling. Based on this setting, the modelling tool can pre-filter all business artefacts, which maybe relevant for the modeller. Furthermore, the tool could also propose reference processes from different business domains. The modeller can use these reference models as an initial version and include the modification for his current business context. In the following section, we will explain the context-driver principle in more detail and show how we can apply it to the most import business artefacts.

4.2 Context-driver Principle

Although the context-driver principle [7] has been applied to business data only, we argue that it can be applied to business process artefacts as well. However, as the structure and information provided in a business process model is much more complex than in business data, our proposal differentiates between the meaning of business artefacts and their relation to each other.

On the one hand, the context-driver principle must be sufficiently discrete in order to enable semantically unambiguous precision. In other words, a semantic meaning of a business process artefact is always unambiguous, when considered in a specific context. For example, a Change Document in an Issue Management process model might contain a field for "bank" which is not precise if this entity is defined without context. "Bank" describes different objects in the industry area of "Finance" and "Marine". Therefore, it is not possible to define only one "bank", which can be used everywhere. Rather, the context in which "bank" is used adds further semantic meaning.

On the other hand, a business process specifies the sequence of activities. In this case, the context-driver principle has to be adapted to support such structural differences. Figure 2 shows a generic example which explains this principle. A business process in context C1 is composed of the activities A1, A2 and A3. In a different context C2, those three activities have the same semantic meaning however there is an additional activity A4, which changes the structure of the process.

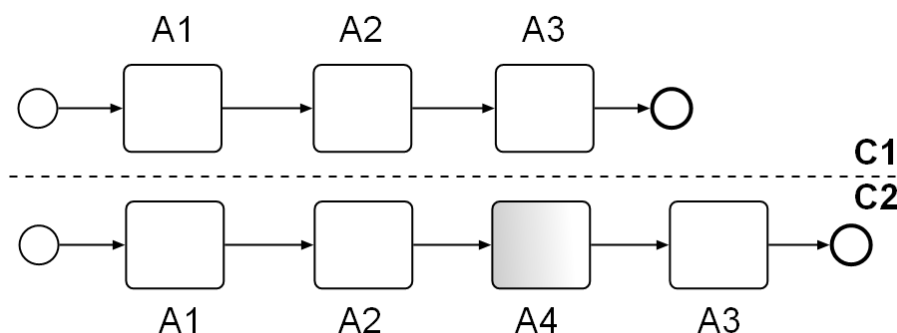


Figure 2 Context-aware Business Process Model

The context-driver principle allows identifying, store, and representing a business process artefact only once while specifying the differences depending on specific context categories (e.g., business process, industry, country, business role, etc.).

In our generic example process, each activity A1, A2, A3 and A4 has only one unique semantic representation in the process repository, however there may be a structural difference (a different predecessor or successor) or even different representations (e.g. synonyms, abbreviations, etc.) depending on the context.

Context defines the environment in which a process artefact is used. The foundation of our work is the Core Components Technical Specification (CCTS) [8] which was proposed by the UN/CEFACT. This specification focuses on the data and information modelling. The idea behind this principle is that all business data follows similar semantic concepts. CCTS provides a common and generic modelling concept for objects and data. With the development of CCTS, the specification already introduced the idea of Context Awareness

among business-related data objects and data types. The concept of Business Contexts and Business Information Entities depicts the concrete characteristic of Context Awareness in CCTS. Business Contexts are classified into eight Context Categories (based on the current version of the specification).

- Business Process
- Product Classification
- Industry Classification
- Geopolitical
- Official Constraints
- Business Process Role
- Supporting Role
- System Capabilities

Introduced in [9], SAP has designed and developed a context-driver principle which is implemented prototypically in the CCTS modelling tool named “Warp 10”. This CCTS Modeller is “an Semantic Web ontology-based data integration, modelling and mapping tool that leverages the semantics of meta data by implementing the semantic-based approach described in ISO 15000-5 Core Component Technical Specification (CCTS)” [9]. The tool focuses on the collaborative, evolutionary, and autonomous data modelling of business artefacts for the application of CCTS in the B2B environment. Figure 3 depicts the simplified meta model of the context-driver approach.

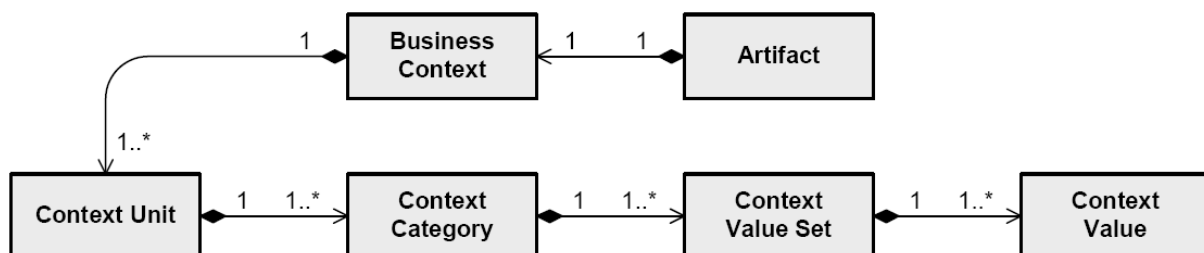


Figure 3 Simplified Meta Model of the Context Driver Principle

Each process artefact is assigned to one Business Context. A Business Context can have one to many Context Units. “A context unit is used to set up the logical conjunction between the different context categories.” Context Units can be compared with Cartesian products of the Context Value Sets of the Context Categories. Context Categories are adopted from CCTS and allow the classification of contexts. Each of the eight Context Categories can refer to one Context Value Set. A Context Value describes a “business situation in an unambiguous and formal way”. [9]

Formally, the Context Driver Principle is based on mathematical sets. Therefore, the mathematical theory of sets is used for the representation for Business Contexts. The following example is meant to be a simple example to give an insight of the usage of the set theory within the approach. The following sets represent three Context Categories. Set *I* (Listing 3.1) represents the industry sector which set contains values such as “Automotive”, “Finance”, “Energy”. Set *G* (Listing 3.2) represents geographical values that can be the ISO codes for countries (ISO 3166) such as “DE” for Germany, “GB” for the United Kingdom, and

“US” for the United States of America. Finally, Set P (Listing 3.3) represents processes that are numbered.

Industry $I = \{\text{Automotive, Finance, Energy}\}$ (Listing 3.1)

Geography $G = \{\text{DE, GB, US}\}$ (Listing 3.2)

Process $P = \{1, 2, 3, 4\}$ (Listing 3.3)

Let us assume, we have defined a process artefact and we want to define a valid Business Context, which specifies that the process artefact is valid in the automotive, and finance sector within Germany and the United Kingdom. It should be also applicable for process 1, 3, and 4. In order to build a valid Business Context, we need to define a Context Unit, which builds a Cartesian product of subsets of the Context Categories. These subsets are listed as set I' , set G' , and set P' ($I' \subseteq I \wedge G' \subseteq G \wedge P' \subseteq P$).

$I' = \{\text{Automotive, Finance}\}$ (Listing 3.4)

$G' = \{\text{DE, GB}\}$ (Listing 3.5)

$P' = \{1, 3, 4\}$ (Listing 3.6)

The valid Business Context which we name BC is a Cartesian product that is shown in Listing 3.7. The defined syntax by [9] of a Business Context is given in Listing 3.8.

$BC = (I' \times G' \times P') = (\{\text{Automotive, Finance}\} \times \{\text{DE, GB}\} \times \{1, 3, 4\})$ (Listing 3.7)

$BC = (I = \{\text{Automotive, Finance}\}; G = \{\text{DE, GB}\}; P = \{1, 3, 4\})$ (Listing 3.8)

A context-driven example is described in [11] which provides an insight of the applied aspects of the Context Driver Principle in Warp 10. For more information about the Context Driver Principle, readers are referred to [9].

4.3 Meta-model

We want to adapt the general approach of CCTS and create a framework for the formalization of business process artefacts. The underlying meta-model for our formalization is depicted in Figure 4, with the Semantic Model Repository as its major component.

A *Model* is described using one or more *Modelling Languages*. Every *Modelling Language* has a number of *Model Elements*. These *Model Elements* are extended by *Labels*. All *Model Elements* can now be linked to the *Semantic Model Repository* and thus every *Model Element* can be linked to a *Business Entity*. A *Business Entity* has a unique semantic meaning and thus is linked to a unique concept of an ontology. In order to make a *Business Entities* meaningful to a person, the framework provides so called *Business Terms* which actually describe a *Business Entity* in a natural language. Both, the *Business Entity* as well as the *Business Terms* might be restricted to a specific *Business Context*. The *Business Terms* are based on a certain *Business Term Grammar*. For example, a business activity

with the concept name "SEND_ORDER" may have the following Business Term representations "send order", "send purchase order", "Auftrag verschicken" or "send PO". In order to be able to support the naming of business terms, our framework relies upon existing *External Dictionaries* such as Wordnet, SAPTerm, etc.

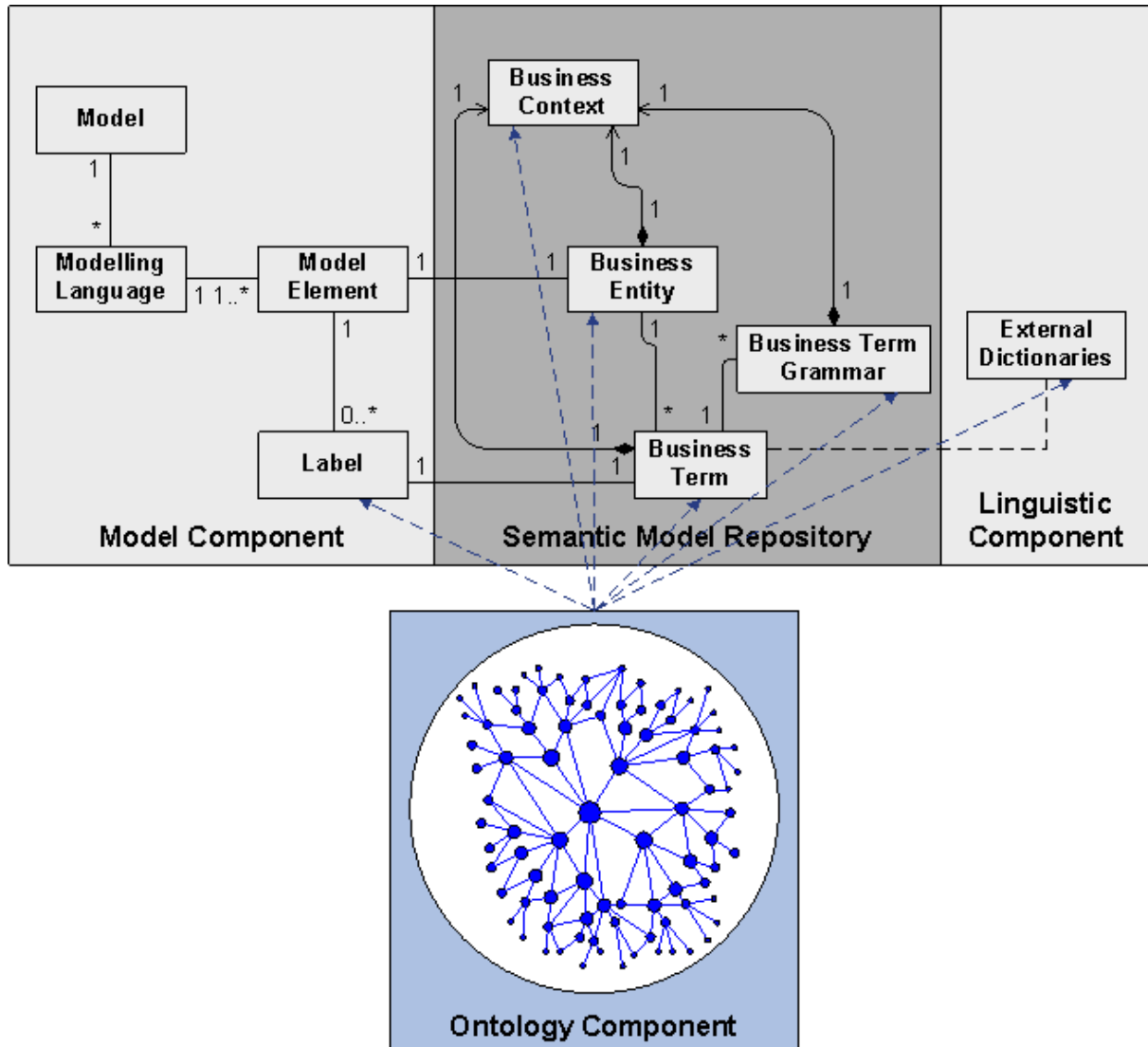


Figure 4 Meta Model

We explain our idea using the previous example (see Figure 1), where the processing of an invoice is modelled using the modelling elements of EPC. A process activity usually represents the processing of a certain resource. In our example, the two different departments use different terminologies to describe the creation of an invoice, namely 'Create Invoice' and 'Bill Creation'. As both activities have the same meaning, both activities will be linked to the same Business Entity which is linked to a unique ontology concept. In our example ontology concept might be CREATE_INVOICE. The abstract Business Entity can have multiple representations in the form of Business Terms. The actual Business Terms might be restricted to a certain linguistic grammar (for example, a verb followed by a space,

followed by a noun, and only be valid in a certain Business Context). In our example the term 'bill' is valid in the sales context, whereas 'invoice' is a valid term in the financial context.

4.3.1 Business Entities

A Business Entity represents a central part of our conceptual framework. In principle, it can be compared with a Business Information Entity in CCTS. [8] However, as by nature the structure of a business process model and its business artefacts is much more complex and has more perspectives than data models, we were only able to reuse the basic idea of CCTS. Business entities represent an abstract concept of business artefacts. They are divided into Basic Business Entities and Complex Business Entities. We have identified the major business artefacts in business process models and categorized them into one of the two groups. This section will detail the individual types of business entities. We do not claim completeness of these types, however, the underlying meta-model of our Semantic Model Repository is flexible and thus additional business entities can be easily added.

Business Entity Business Entities represent objects of the business world. In our work, these are especially business-related objects which are used in business process models. Business Entities are divided into Basic Business Entities and Complex Business Entities.

- **Basic Business Entity** Basic Business Entities are atomic and represent atomic aspects that are needed within business process modelling.
 - **Role** *Roles* can either be persons or groups of persons who are involved in the operation of business processes, particularly in *Activities* as they are the only executing part of *Processes*.
 - **Resource** *Resources* can be anything which is needed or used within an *Activity* and therefore within a *Process*. Examples of *Resources* are documents, pieces of information, messages, information systems, databases, but also materials, etc.
 - **Action** An *Action* is the process of doing something. *Action* are described in a basic form. They define in combination with a *Resource* the core component of an *Activity*.
 - **Predicate** A *Predicate* makes a semantic statement about a subject to which it is related. *Predicates* are used in structures that connect *Business Entities* to each other. For example, they define the relation that a certain *Resource* is an input for a certain *Activity*.
 - **State** A *State* defines the condition of a *Complex Business Entity* in which it is embedded.
 - **Goal** A *Goal* is the representation of an objective which fulfilment is achieved through the execution of one or more services. *Goals* can be linked to activities, processes or workflow templates.
 - **Service** A *Service* is a software component which can be executed remotely and which fulfils certain *Goals/a Goal*. *Services* can be linked to *Activities*.
- **Complex Business Entity** *Complex Business Entities* are variable structures which connect *Basic Business Entities* to each other. These connections build a semantic meaning, which represents the *Complex Business Entity* itself.
 - **Activity** *Activities* represent the executing part of business processes. *Activities* are performed by *Services* or *Roles* and require *Resources*. The core component of every *Activity* is an *Action*, which processes a *Resource*. After an *Activities* is

executed by, it creates a new state. Roles, Resource, Goals and Services can have several relations to Activities. These relations are marked by Predicates which express the semantic meaning of the relations. Relations which do not define the semantic meaning of the actual Activities are called *Activity Structures*.

- **Workflow Template** A Workflow Template contains a predefined subset of Activities. These Activities can be arranged in various orders and paths which define the sequential execution flow within a Process. Processes can only contain Activities and other Processes, which build then sub-processes of the embedding Process.
- **Process** A Process contains a set of Activities. These Activities can be arranged in various orders and paths which define the sequential execution flow within a Process. Processes can only contain Activities and other Processes, which build then sub-processes of the embedding Process. In addition, Process can also contain Workflow Templates.
- **Condition.** An condition is an axiom or any logical expression that states some fact about a set of basic or complex business entities.

Each Business Entity concept (e.g. “CREATE_INVOICE”) has only one instantiation in the repository. As the instantiation is further defined using a Business Context (see Section 4.2) which describes the valid space for it, the process model can be reflected using such context assignments.

5. Language Symbols, Process Patterns, and Workflow Templates









Section 3 has presented our design principle of the lightweight process modelling language. In the following sections, we provide symbols used in the language in Section 5.1. We provide some process modelling patterns and workflow templates, which should help to identify the required level of detail in the case studies in Section 5.2 and Section 5.3. Goals and template processes are introduced in Section 5.4. A meta-model for lightweight process modelling language is presented in Section 5.5. Finally, we summarize and highlight some issues related with the current version of the language.

This version of the lightweight modelling language focuses on patterns and is restricted to the process (i.e. control-flow) perspective. All process patterns come from workflow patterns [12]. For each selected pattern, we provide examples of real cases in small set of BPMN notation. Some process modelling fragments in BPMN are specified as workflow templates. A YAWL [13] version of process patterns and templates can be finding from Annex A.

5.1 Symbols Used in Lightweight Process Modelling Language

Here we have adopted seven notations from BPMN and added two goal related notations. Research shows that the average subset of BPMN used in these models consists of just nine different symbols [25]. In future, we may extend the selection of symbols. It depends on the requirements from WP7, WP8, and WP9.

Table 1 Symbols used in Lightweight Process Modelling Language.

	Starts a process flow		Ends a process flow
	An activity is a unit of work, the job to be performed		Sequence Flow defines the execution order of activities
	Exclusive Gateway When splitting, it routes the sequence flow to exactly one of the outgoing branch on conditions. When merging, it awaits one incoming branch to complete before triggering the outgoing flow		Parallel Gateway When used to split the sequence flow, all outgoing branches are activated simultaneously. When merging parallel branches it waits for all incoming branches to complete before triggering the outgoing flow
	Inclusive Gateway When splitting, one or more braches are activated based on branching conditions. When merging, it awaits all active incoming branches to complete.		
	Atomic Activity Goal Atomic goals are those that are associated with a single concrete activity,		

involving just one step of computation.



Composite Activity Goal

Composite goals are those which fulfilment involves the completion of other simpler subgoals. In practice, this means that there is a process associated with the complex goal, process that describes how the described capability is realized either by achieving other goals, or by invoking concrete services.

We will use above language symbols to describe process patterns and workflow templates.

5.2 Description of Process Patterns

This section comprises an introduction of the most important process patterns. Process patterns are used in order to simplify business process modelling. Patterns can accelerate the process of designing a solution and minimize designs made from scratch.

We choose to use the most frequently used workflow patterns from [12] as our *process pattern*. The patterns range from very simple to very complex and cover the behaviours that can be captured within most business process models. In order to facilitate the understanding of the patterns in this deliverable we will use the BPMN for their description.

The average subset of BPMN used in most of the existing process models consists of just nine different symbols [25]. In future, we may extend our set of symbols according to the requirements out of WP7, WP8, and WP9.

Pattern 1 (Sequence)

Description An activity/task should await the completion of another activity within the same case before it can be scheduled.

Example

The activity/task *select_winner* is followed by the activity *notify_outcome*.

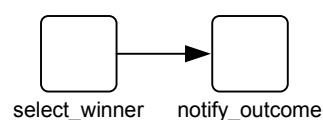


Figure 5 Sequence in BPMN.

Pattern 2 (Parallel Split)

Description The divergence of a branch into two or more parallel branches each of which execute concurrently.

Example

An example of Pattern 2 could be in the context of an application process where after creating a short-list of candidates, referee reports need to be obtained and interviews need to be held.

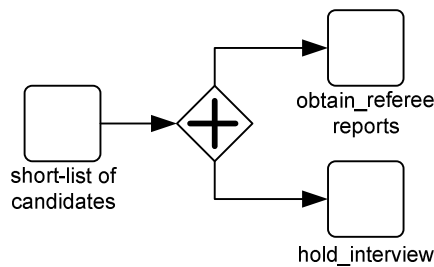


Figure 6 Parallel Split in BPMN.

Pattern 3 (Synchronization)

Description Two or more incoming branches converge into a single subsequent branch. The thread of control is passed to the subsequent branch when all input branches have been enabled.

Example

In the context of an application process, a decision for a particular candidate can only be made once their referee reports have been received and they have been interviewed.

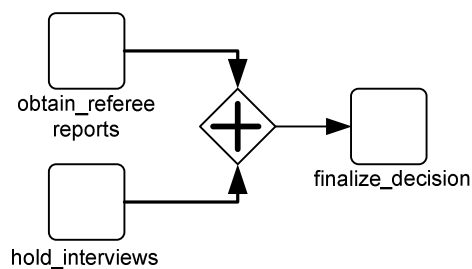


Figure 7 Synchronization in BPMN.

Pattern 4 (Exclusive Choice)

Description A branch diverges into two or more branches. Out of two or more outgoing branches, one branch is chosen. When the incoming branch is enabled, the thread of control is immediately passed to precisely one of the outgoing branches based on the outcome of a logical expression associated with the branch.

Example

An example of Pattern 4 consider the case where purchase requests exceeding \$10,000 are to be approved by head office, while purchase requests not exceeding this amount of money can be approved by the regional offices.

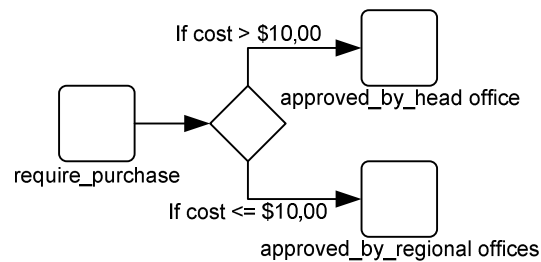


Figure 8 Exclusive Choice in BPMN.

Pattern 5 (Simple Merge)

Description One of preceding branches completes. Two or more branches converge into a single subsequent branch. Each enablement of an incoming branch results in the thread of control being passed to the subsequent branch.

Example

An example of Pattern 5 could be in the context of an application process where after finalizing decisions of rejection or approval, a report is issued.

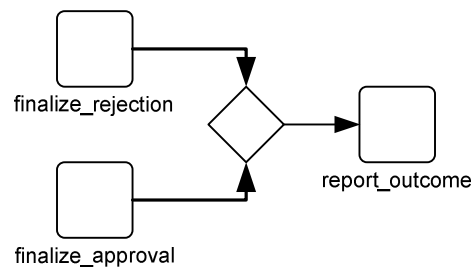


Figure 9 Simple Merge in BPMN.

Pattern 6 (Multi-choice)

Description Out of several branches, a number of branches are chosen based on user input.

Example

After the execution of activity *determine_teaching_evaluation*, execution of activity *organize_student_evaluation* may commence as well as execution of activity *organize_peer_review*. At least one of these two activities is executed, possible both.

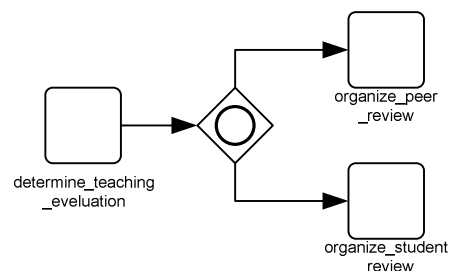


Figure 10 Multi-choice in BPMN

Pattern 7 (Synchronizing Merge)

Description A form of synchronisation where execution can proceed if and only if one of the incoming branches has completed and from the current state of the workflow it is not possible to reach a state where any of the other branches has completed.

Example

Consider again the example presented in Pattern 6 (Multi-choice). After activities *organize_student_evaluation* and *organization_peer_review* have finished, activity *interpret_results* could be scheduled. This activity should only await completion of those activities that were actually executed and itself be performed once.

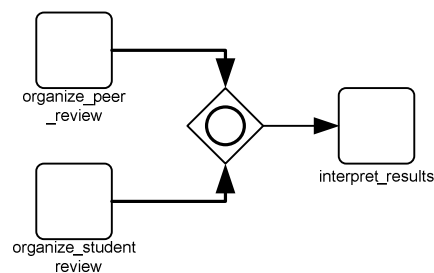


Figure 11 Synchronizing Merge in BPMN.

Pattern 8 (Multi- Merge)

Description It will execute the activity/task involved as many times as its incoming branches signal completion. Each enablement of an incoming branch results in the thread of control being passed to the subsequent branch.

Example

The *lay_foundations*, *order_materials* and *book_labourer* activities occur in parallel as a separate process branches. After each of them completes the *quality_review* activity is executed for each time one of the three tasks completes.

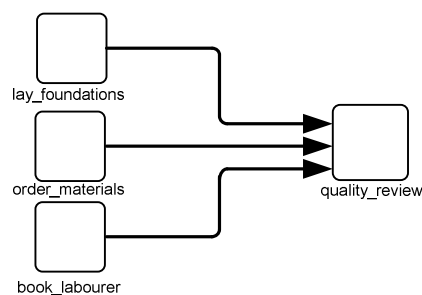


Figure 12 Multi-Merge in BPMN.

Pattern 9 (Discriminator)

Description It provides a form of synchronization for an activity where out of a number of

incoming branches executing in parallel, the first branch to complete initiates the activity. When the other branches complete they do not cause another invocation of the activity.

Example

When handling a cardiac arrest, the *check_breathing* and *check_pulse* tasks run in parallel. Once the first of these has completed, the *triage* task is commenced. Completion of the other task is ignored and does not result in a second instance of the *triage* task.

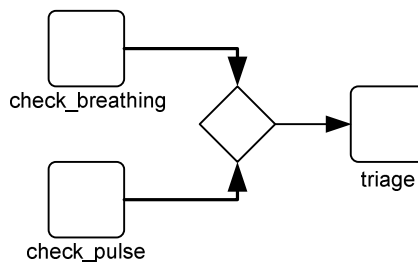


Figure 13 Discriminator in BPMN.

5.3 Workflow Templates

Applying process patterns at the process modelling stage, end users can obtain support if applying a pattern causes a modelling error. The sequence of applied patterns can be traced during process editing time. However, the process patterns are fine-grained and not sufficiently enriched with information on the context and consequences to represent a reusable solution. Therefore, we introduce workflow templates that are different combinations of process patterns. The processes of each workflow template represented are sound. Certain workflow templates can be enriched with the information that they are valid for different domains, i.e. business context.

Never wait, executed every time

Consider booking of a business trip as an example, it starts with an OR-split *register* which enables task *booking_flight*, *reserving_hotel* and/or *renting_car*, activity *pay* is executed for each time one of the three tasks (i.e., *booking_flight*, *reserving_hotel* and *renting_car*)

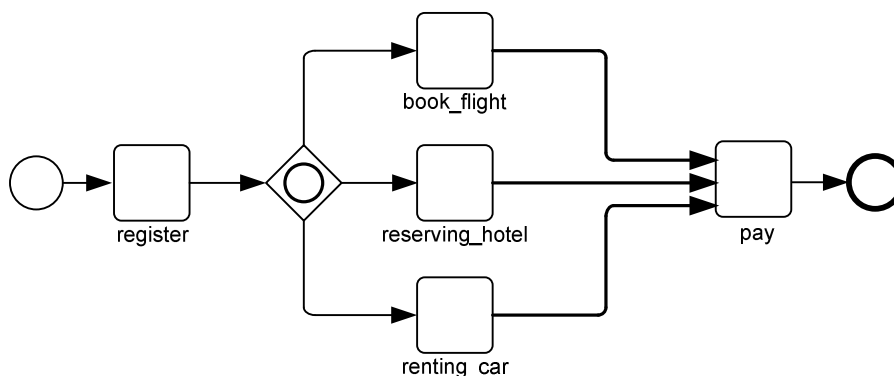


Figure 14 Task *pay* is executed each time one of the three preceding task completes

Wait for all to come

Figure 16 is similar but combines the individual payments into one payment. Therefore, it waits until each of the tasks enabled by *register* completes. If only a flight is booked, there is no synchronisation. However, if the trip contains two or even three elements, task *pay* is delayed until all have completed.

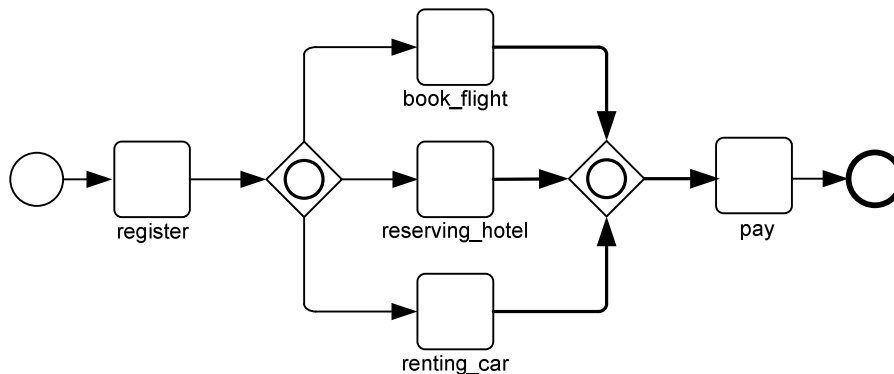


Figure 15 Task *pay* is executed only once, i.e., after all started tasks have completed

Wait for first to come and ignore others

Figure 17 enables all three activities *booking_flight*, *reserving_hotel* and *renting_car*, activity *pay* is executed after the first task is completed. After the payment all running tasks are cancelled.

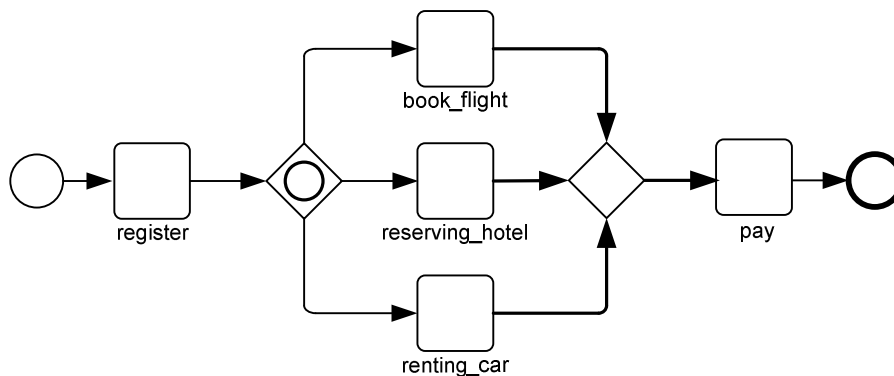


Figure 16 Task *pay* is executed only once, i.e., after the first task has completed.

5.4 Goals and Template Processes

The activities that compose the lightweight processes that we have presented so far are concrete. As we restrict ourselves to service-oriented environments, this concrete activities represent the execution of concrete services. These services must be selected to address particular needs in a particular context; and hence, the process must be configured on a per case basis. Furthermore, those independent services are composed in a certain order to satisfy the various global process business goals. In this section, we introduce a new component in processes, goals, that will allow us to define more abstract and user-friendly processes.

As defined in [16], goals are the representation of an objective which fulfilment is sought through the execution of a (possibly complex) service. We can see goals from two different perspectives.

- **From the provider perspective**, they are like abstractions over similar services.
- **From the client side** they represent the user requirement, what the user wants to be carried out by the service-oriented system.

The main advantages of the goal-driven approach to the definition of processes are:

- **Role separation.** One of the big improvements of the use of goals is that they allow the separation of the user specific context and vocabulary from the provider, as they probably will not be the same.
- **Capability based invocation.** The inclusion of goals allows us to abstract us away from concrete services and describe process steps in terms of their desired capability. As described in [19], building on the principle above users can focus on selecting the goals they want to be achieved; and the system is the one in charge of selecting the appropriate underlying services.
- **Process parameterization.** Finally, the inclusion of goals facilitates the needed degree of freedom to define the parameterized process templates that we can reify later on to be applied in different situations. Goals become placeholders that permit some degree of flexibility, since if we wish to adapt processes to different situations, we will need some degree of freedom in its definition. With this purpose, we define parameterized process templates, which we portray in Figure 17. Parameterized process templates are composed by the elements that we have enumerated up to now (i.e. task, conditions, patterns, etc.), mixed with goals.

We define two types of goals, namely atomic goals and composite goals (represented in Figure 17 and Figure 18).

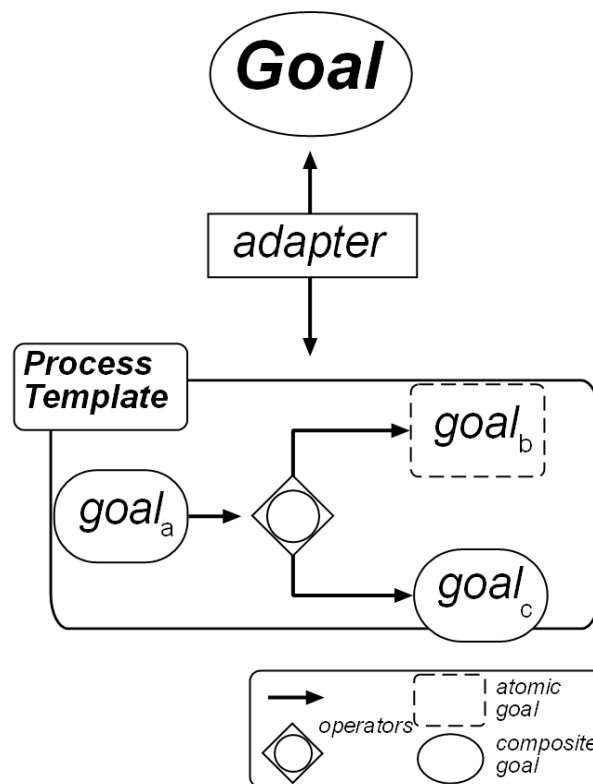


Figure 17 Composite Goals and Process Templates Graphical Representation.

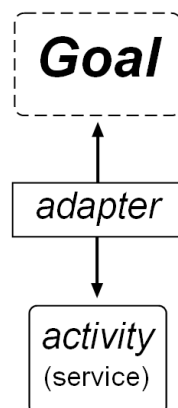


Figure 18 Atomic Goal Graphical Representation.

- **Atomic goals** are those that are associated with a single concrete activity, involving just one step of computation hence.
- **Composite goals** are those which fulfilment involves the completion of other simpler subgoals. In practice, this means that there is a process associated with the complex goal, process that describes how the described capability is realized either by achieving other goals, or invoking concrete services.

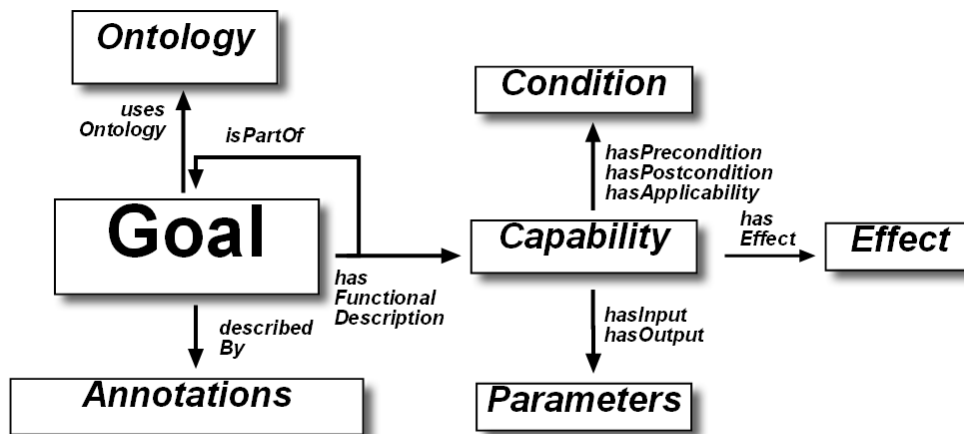


Figure 19 Goal Main Components.

In Figure 19 we depict the main knowledge components of a goal, which we have tried to align as much as possible with the WSML-Lite specification. Firstly, a goal can be associated with other goal, using an inclusion relationship. A goal is also associated with a set of meta-knowledge tags about the goal, for which we choose the Annotation concept defined in the WSMO ontology [16]. Finally, regarding the competence of the goal, our approach uses the extension that appears in IRS III, defining goals with inputs and outputs in order to facilitate capability-based invocation [19] as we have previously stated. Regarding the formalism used to define the axioms that compose the competence of a goal (pre/post condition, effect and applicability) we are initially agnostic. We restrict ourselves to a knowledge-level definition and we will not prescribe any concrete language nor reasoning mechanisms.

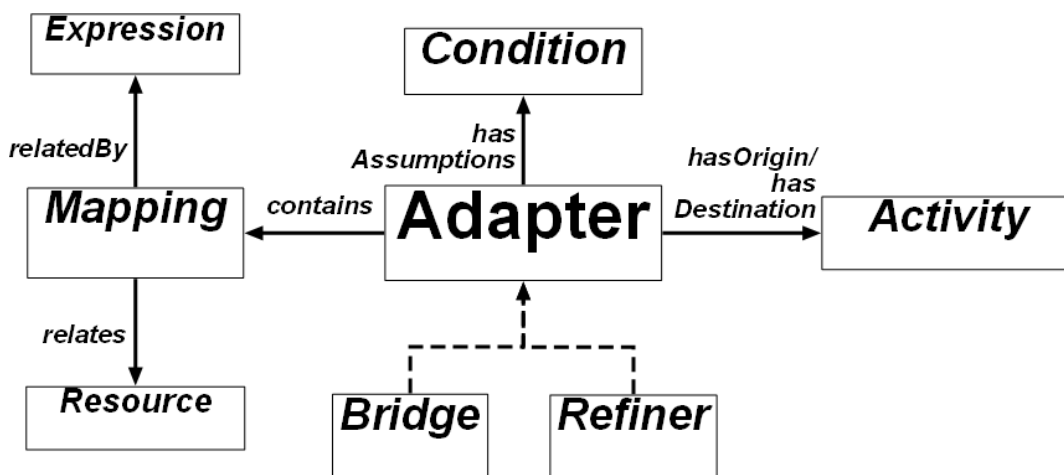


Figure 20 Adapter main components

Goals, processes templates and services are connected via adapters. Adapters [18] allow the independent specification of problem definitions. This definition, somehow too general, is further specified in [17] where adapters gained their status as first class knowledge

components. Fensel describes that the construction of knowledge-based systems from reusable elements requires the adapters that adjust them to the application-specific circumstances, among which context-dependent issues are of great importance for us. To sum up, adapters will on the one hand avoid terminological mismatches between goals and lightweight process templates; and on the other hand, they will state explicitly the domain and context applicability assumptions that assure that the process template provides the functionality depicted by the goal. As represented in Figure 20 the adapter contains a set of mappings among resources, resources that ideally are related with each of the activities that the adapter associates. As important as these mappings, are the assumptions that the adapter makes about the domain and context of applicability of the adapter.

We include adapters and we do not use mediators since we consider adapters as a simplification of WSMO mediator. Mediators are services, and therefore active entities, that can accomplish several levels of mediation (see [16]); whilst adapters are just passive expressions that relate activities with goals. They just assert small ontological-level mediation, and make explicit assumptions about the context and domain where the relationship between activities holds.

Finally, we believe that It is of great importance to stress that these goals and adapters definitions are carried out by domain experts with the aid of knowledge representation experts, not by non-expert users. Users make effective use of goals, but we cannot assume that they will be able to define such kind of knowledge components.

5.5 Meta-model of the Lightweight Process Modelling Language

In this section, we introduce a meta-model for the lightweight process modelling language (shown in Figure 21). The meta-model depicts the relationships between all the elements in a process model. Elements, such as “Start”, “End”, “Activity”, “SequenceFlow”, “ExclusiveGateway”, “ParallelGateway”, “InclusiveGateway”, “AtomicActivityGoal”, and “CompositeActivityGoal”, have been described in Section 5.1. An abstract concept “Gateway” is used as a super set of three gateways, i.e. “ExclusiveGateway”, “ParallelGateway”, and “InclusiveGateway”. An “AtomicActivity” is the smallest unit of work. A “CompositeActivity” consists of several other activities, either atomic or composite. An “Activity” is either an “AtomicActivity” or a “CompositeActivity”. A “ProcessPattern” is introduced in Section 5.2. A “WorkflowTemplate” is defined in Section 5.3. An “AtomicActivityGoal” and “CompositeActivityGoal” are explained in Section 5.4. An “ActivityGoal” consists of either an “AtomicGoal” or a “Composite Goal”. “FlowObject” in the figure represents a process model. We separate “NonSequenceFlowObject” from “SequenceFlow”. It ensures the alternating ordering of SequenceFlow (arrows) and other objects and thus avoids the situation two “SequenceFlow” elements directly linking to each other.

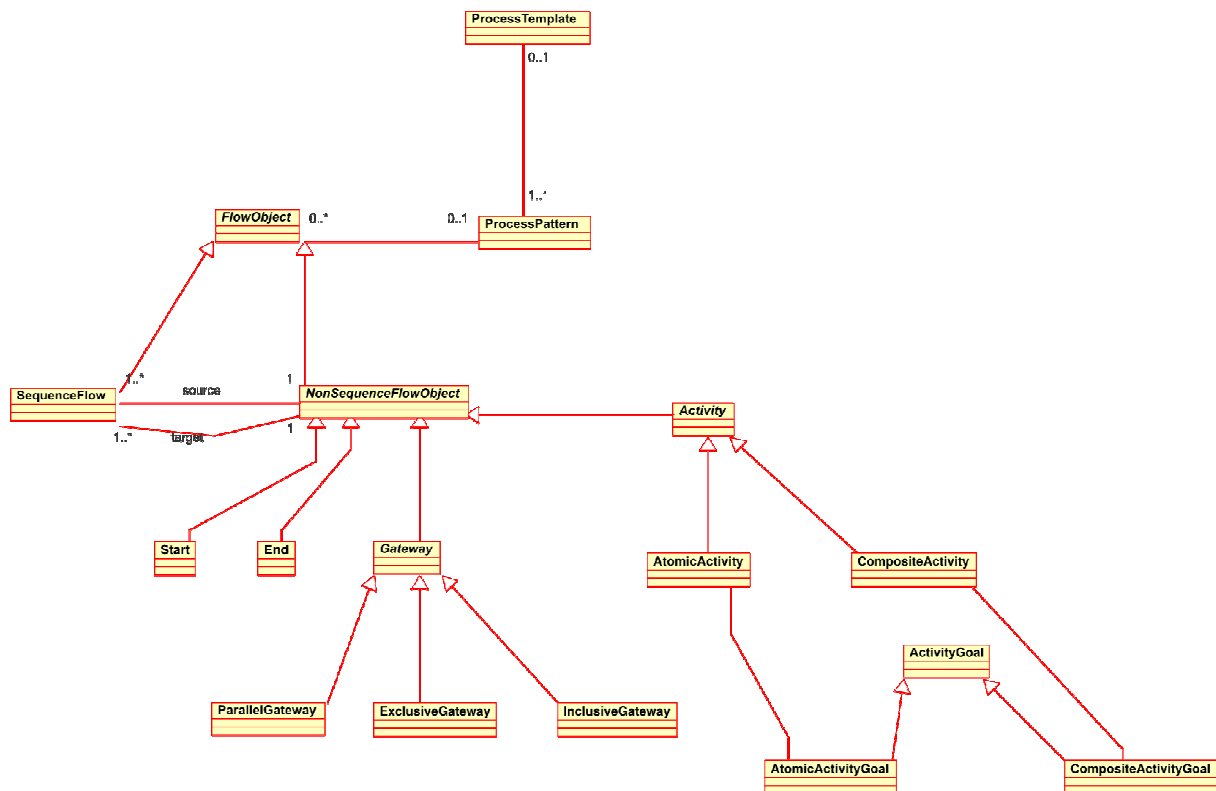


Figure 21 Meta Model of the Lightweight Process Modelling Language

5.6 Summary and Remarks

In this chapter, we introduced the symbols used in the lightweight process modelling language. We provide some process modelling patterns and workflow templates, which should help users to minimize design made from scratch. We further provide activity goals as unbound activities that are bound to a particular service at runtime. Finally, a meta-model of the lightweight process language is presented.

In the design of a lightweight executable process modelling language, it is important to keep a balance between simplicity of use and expressive power of the language. We keep the logic expression of a control flow as simple as possible, but it is enough for execution. It is possible for a process editor (T2.6) to support a subset of the language and to use alternative means to get modelling information in more flexible ways such as through the provision of advanced guidance during modelling activities. In the process editor, the end users can specify a process model at a high abstraction level. A mapping from a high-level process model into an executable model is then needed. In this deliverable, we mainly concentrate on the lightweight process modelling language (executable one) itself. The symbols used in the process editor for creating a process model are not part of this deliverable. The advanced guidance during modelling activities is not a part of the deliverable either.

Data flow is not yet support by this version of the language. Data processing functionalities of process data will be supported after refining the current concepts of the lightweight, context-aware process modeling language and further experience with the usage of the language on the cases presented in the use case deliverables.

Making our graphical based process modelling language available for executable processes is not yet addressed in this deliverable. We have considered various languages such as XPDL [27] and WSBPEL [28] as targets for a representation of the language that is supported by process execution engines, as well as the semantic matching of those languages with our language. However, the final decision of storage language should also be based on agreements with T6.4 and T6.5. Therefore, we will work together closely with T2.6, T6.4 and T6.5.

6. Language Evaluation

In this section we provide an overview of the language evaluation process, which is based on Lab experiments i.e., experiments that involves real end users carrying out several tasks. Besides considering Lab experiments, we will first consider some evaluations based on:

- Heuristic evaluation [2] i.e., it is a method for discovering usability issues in the user interface design. The following steps are usually followed in heuristic evaluation:
 - Brief the experts about what to do
 - Every expert independently evaluates the system for 1-2 hours as follows:
 - Go through the system to get the feel of the product
 - Go through the system for a second time but focus on specific features of the product
 - Debriefing session in which experts work together to categorise the problems.
- Cognitive Walkthrough (i.e., usability inspection method performed by an evaluator who walks through a pre-planned scenario to identify usability problems within a system [1]),
- focus groups [3] i.e., a group of people are asked about their attitude towards a product or system. Questions are asked in an interactive style where participants are free to talk with other participants. The idea is to capture the information about users' need and issues related to the system.

Details of such approaches are considered in a separate document D2.5.1. In the following, we will focus on the Lab experiments based evaluation plan.

6.1 Lab Experiments

Empirical experiments will be carried out to test the final language, which will serve as a basis to model complex services such as composite services (i.e., composition of Web services) in SOA4ALL. These experiments involve real end users carrying out several tasks. From these experiments, we, as evaluators, then analyse the results of the experiment to check whether the language proposal supports the users in accomplishing their tasks and identify usability problems.

6.2 Feedback from Use Cases

As we have exposed the definition of the language will evolve; our approach to the construction of the lightweight process modelling language will be heavily driven by the lab experiments and by use cases requirements. In consonance, we have sent the first version of process patterns and workflow templates (see to Section 4 and Annex A) to WP7, WP8, and WP9. The feedback from WP8 in terms of expected usage of our initial set of patterns is summarized in Table 2.

Table 2 Feedback from WP8.

Pattern 1	Yes	Pattern 2	Yes	Pattern 3	Yes
-----------	-----	-----------	-----	-----------	-----

Pattern 4	Yes	Pattern 5	Yes	Pattern 6	Yes
Pattern 7	Yes	Pattern 8	Yes	Pattern 9	Maybe

The feedback from WP9 is summarized as new requirements/comments and the actions that we have already taken to address them.

- *“The typical user of the WP9 eCommerce framework will have a hard time to distinguish the different versions of split and join types (and how tokens are processed in general)”.*

Answer: They may choose a workflow template which can do the work from the text description and goals specifications and run it.

- *“The service broker combines services to provide “service bundles” to the end users. These service bundles consist of simple processes, for example the combination of fraud detection and address check services with the actual payment service”.*

Answer: Templates processes thanks to the use of goals can be easily catalogued and used as an off-the-shelf solution to the service broker users.

- *“The users of the eCommerce application would create suitable compositions of services, based on workflow templates, which simulate the usual order process for a customer of the eCommerce application. For these workflow templates we need a very simple formalism, and the ability to exchange some of the activities with real services. ”*

Answer: Being an executable process model, the model itself can not be very simple. However, we can try to hide the complexity away from users. We do allow end users to specify some constraints, such as a constraint that certain tasks or activities in the process model must be performed by certain pre-required Web services.

- *“So while we could have a more detailed way for process composition on the service broker side, the composition for the end users in the eCommerce platform should be even more abstract than presented in your document. Thus, while it is important to support these different patterns, we should find a way to “hide” some notational details from the user... ”*

Answer: We certainly hide some details. Uses can specify “goal” for their needs.

- *“Moving to BPMN is okay from my point of view, I even think the notation is a bit more intuitive than in YAWL.”*

Answer: We use BPMN notation.

7. Conclusions

In the context of WP6 of the SOA4All project, the term service construction mainly refers to the modelling of processes and the execution of composite services and processes in a lightweight manner. This should enable different groups of end users to build new services and processes according to their specific needs.

In this deliverable, we have studied the requirements of lightweight, context-aware process modelling language from WP7, WP8, and WP9 and provided our methodology of requirements acquisition. After the requirement analysis, we presented the design principles of the process modelling language: context-awareness, reusability, and flexibility. The context-awareness principle aims to alleviate semantic ambiguity by matching terms from a common business ontology. Further, the context-driven principle allows identification, storage, and representation of a process artefact/model only once while specifying the differences depending on specific context categories. It also contributes to reusability. In order to simplify process modelling, process models must be highly reusable, favouring process flexibility and minimizing design made from scratch. We choose to use well-know workflow patterns as our process patterns because of their coverage of behaviours in process models and their formal foundations. These formal foundations will bring value during process design time, process deployment time and run time. Further, we have also provided a summary of language evaluation in terms of an evaluation plan and feedback from related work packages in SOA4All.

Future research of T6.3 will be performed in cooperation with T2.6, T6.4, T6.5, WP7, WP8, and WP9 to refine the language (such as supporting data-flow), to detail lightweight process methods, and to solve other implementation related issues. As the delivery of the latest version of D7.3 and D8.3 has been postponed until M13, this version of language specification cannot completely reflect the final requirements of WP7 and WP8.

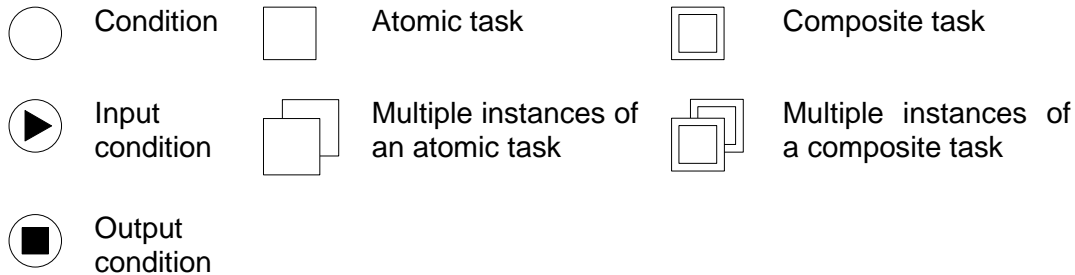
8. References

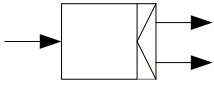

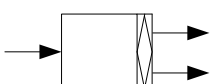
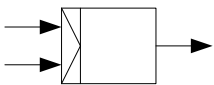
- [1] Gartner Group. (2008). Magic Quadrant for Business Process Analysis Tools, 2H07-1H08 (Gartner Core Research Note G00148777). Stamford, CT: Gartner, Inc., 2008.
- [2] Jan Mendling and Jan Recker. Towards systematic usage of labels and icons in business process models. In CAiSE 2008 Workshop Proceedings - Twelfth International Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD 2008), 2008.
- [3] Rolland, C.; Pernici, C. Thanos (June 1998). A Comprehensive View of Process Engineering. Proceedings of the 10th International Conference CAiSE'98, B. Lecture Notes in Computer Science 1413. Pisa, Italy: Springer.
- [4] C. Wharton et al. "The cognitive walkthrough method: a practitioner's guide" in J. Nielsen & R. Mack "Usability Inspection Methods" pp. 105-140
- [5] Jakob Nielsen. Ten Usability, from http://www.useit.com/papers/heuristic/heuristic_list.html
- [6] J. Nielsen "Usability Engineering" pp.214-216, Academic Press, 1993
- [7] Gunther Stuhec. How to Solve the Business Standards Dilemma the Context Driven Business Exchange. SAP Developer Network Article, 20th October 2005, <https://www.sdn.sap.com/irj/servlet/prt/portal/prtroot/docs/library/uuid/a6c5dce6-0701-0010-45b9-f6ca8c0c6474>
- [8] UN/CEFACT United Nations Centre for Trade Facilitation and Electronic Business. Core components technical specification - part 8 of the ebXML framework, November 2003. Version 2.01.
- [9] Gunther Stuhec and Mark Crawford. Accelerate your Business Data Modeling and Integration Issues by CCTS Modeler Warp 10. Technical report, SAP AG, November 2007. <https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/00435406-4b7e-2a10-b28a-b7143af53e88>; Date of Retrieval: 3 January 2009.
- [10] Gunther Stuhec and Huiming Yu. Context Driven Approach. Technical report, SAP AG, December 2007. Status: Draft, http://www.unstandards.org:8080/download/attachments/3801833/Contribution++SAP++ContextDrivenApproach_1p10.pdf; Date of Retrieval: 30 September 2008
- [11] Gunther Stuhec. Using CCTS Modeler Warp 10 to Customize Business Information Interfaces. Technical report, SAP AG, November 2007. <https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/70d6c441-507e-2a10-7994-88f6f769d6e8>; Date of Retrieval: 3 January 2009.
- [12] W.M.P van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns..Distributed and Parallel Databases, 14(3), pages 5-51, July 2003.
- [13] W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. Information Systems , 30(4):245-275, 2005.
- [14] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Professional Computing Series. Addison Wesley, Reading, MA, USA, 1995.
- [15] D. Riehle and H. Z"ullighoven. Understanding and Using Patterns in Software Development. Theory and Practice of Object Systems, 2(1):3–13, 1996.

-
- [16] WSMO Web Service Modeling Ontology (WSMO) WSMO Final Draft 21 October 2006 <http://www.wsmo.org/TR/d2/v1.3/>
- [17] Fensel, D. (1997) The Tower-of-Adapter Method for Developing and Reusing Problem-Solving Methods. In R. Benjamins and E. Plaza (Editors). Knowledge Acquisition, Modeling, and Management. Proceedings of the 10th European Workshop, EKAW '97. Lecture Notes in Artificial Intelligence 1319, Springer-Verlag.
- [18] Fensel, D. and Groenboom, R. (1997). Specifying Knowledge-Based Systems with Reusable Components. In Proceedings of the 9th International Conference on Software Engineering & Knowledge Engineering (SEKE-97), Madrid, Spain, June 18-20.
- [19] Cabral, L., Domingue, J., Galizia, S., Gugliotta, A., Norton, B., Tanasescu, V. and Pedrinaci, C., IRS-III: a broker for semantic web services based applications. In: Proceedings of the 5th International Semantic Web Conference.
- [20] S.A. White. Process Modelling Notations and Workflow Patters, www.bpmn.org. Jan. 2004
- [21] Jog Roj, Martin Owen. BPMN and Business Process Management, www.bpmn.org. Sept. 2003
- [22] Buschmann, F. , Henney, K., Schmidt, D.C. Past, Present, and Future Trends in Software Patterns. IEEE Software 24 (7/8), 31-37 (2007)
- [23] John Medicke and Doug McDavid, Patterns for Business Process Modeling. Business Integration Journal 1, 32-35 (2004)
- [24] Hanh Nhi Tran, Bernard Coulette, Bich Thuy Dong. Broadening the Use of Process Patterns for Modeling Processes. In: Proc. SEKE, Knowledge Systems Institute Graduate Schools, pp. 57-62 (2007).
- [25] Michael zur Muehlen. How much BPMN do you Need? BPM Research, <http://www.bpm-research.com/2008/03/03/how-much-bpmn-do-you-need/>
- [26] EU. The PICTURE Project. <http://www.picture-eu.org/>
- [27] XDPL 2.1.
http://www.wfmc.org/index.php?option=com_docman&task=cat_view&gid=42&Itemid=72
- [28] WSBPEL, OASIS Web Services Business Process Execution Language (WSBPEL) TC: <http://www.oasis-open.org/committees/wsbpel/>
- [29] SOA4All D6.1.1, State of the Art Report and Requirements for Service Construction.
- [30] SOA4All D2.6.1, Specification of the SOA4All Process Editor.
- [31] R Shapiro, A Comparison of XPDL, BPML and BPEL4WS. Cape Visions.
<http://xml.coverpages.org/Shapiro-XPDL.pdf>
- [32] Frauke Paetsch, Armin Eberlein, and Frank Maurer, "Requirements Engineering and Agile Software Development," Enabling Technologies, IEEE International Workshops on, vol. 0, no. 0, pp. 308, Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.

Annex A. Process Patterns and Templates in YAWL

Symbols used in YAWL



Split Types		
Name	Symbol	Description
And-split task		The AND-Split is used to start a number of new pieces of work simultaneously. It can be viewed as a specialisation of the OR-Split, where work will be triggered to start on all outgoing flows.
XOR-split task		The XOR-Split is used to trigger only one outgoing flow. It is best used for automatically choosing between a number of possible exclusive alternatives once a task completes.
OR-split task		The OR-Split is used to trigger some, but not necessarily all outgoing flows to other tasks. It is best used when we won't know until run-time exactly what concurrent resultant work can lead from the completion of a task.
Join Types		
Name	Symbol	Description
AND-join task		A task with an AND-Join will wait to receive completed work form all of its incoming flows before beginning. It is typically used to synchronise pre-requisite activities that must be completed before some new piece of work may begin.

XOR-join task		Once any work has completed on an incoming flow, a task with an XOR-Join will be capable of beginning work. It is typically used to allow new work to start so long as one of several different pieces of earlier work have been completed.
OR-join task		The OR-Join ensures that a task waits until all incoming flows have either finished, or will never finish. OR-Joins are “smart”: they will only wait for something if it is necessary to wait. However, understanding models with OR-joins can be tricky and therefore OR-joins should be used sparingly.



Table 3 Symbols used in YAWL

Description of Process Patterns in YAWL

Pattern 1 (Sequence)

Description An activity/task should await the completion of another activity within the same case before it can be scheduled.

Example

The activity/task *select_winner* is followed by the activity *notify_outcome*.

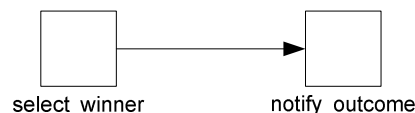


Figure 22 Sequence in YAWL.

Pattern 2 (Parallel Split)

Description The divergence of a branch into two or more parallel branches each of which execute concurrently.

Example

An example of Pattern 2 could be in the context of an application process where after creating a short-list of candidates, referee reports need to be obtained and interviews need to be held.

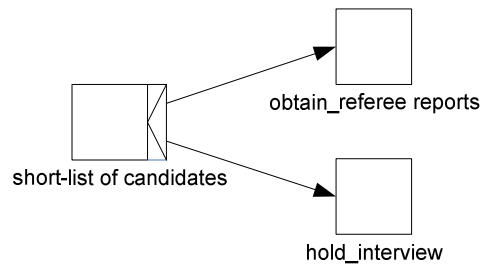


Figure 23 Parallel Split in YAWL.

Pattern 3 (Synchronization)

Description Two or more incoming branches converge into a single subsequent branch. The thread of control is passed to the subsequent branch when all input branches have been enabled.

Example

In the context of an application process, a decision for a particular candidate can only be made once their referee reports have been received and they have been interviewed.

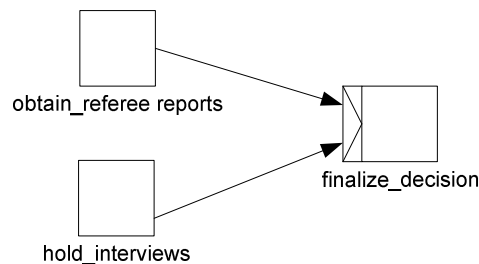


Figure 24 Synchronization in YAWL.

Pattern 4 (Exclusive Choice)

Description A branch diverges two or more branches. Out of two or more outgoing branches, one branch is chosen. When the incoming branch is enabled, the thread of control is immediately passed to precisely one of the outgoing branches based on the outcome of a logical expression associated with the branch.

Example

An example of Pattern 4 consider the case where purchase requests exceeding \$10,000 are to be approved by head office, while purchase requests not exceeding this amount of money can be approved by the regional offices.

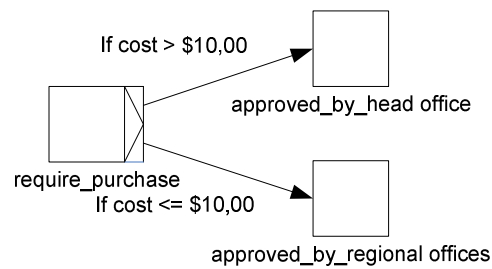


Figure 25 Exclusive Choice in YAWL.

Pattern 5 (Simple Merge)

Description One of preceding branches completes. Two or more branches converge into a single subsequent branch. Each enablement of an incoming branch results in the thread of control being passed to the subsequent branch.

Example

An example of Pattern 5 could be in the context of an application process where after finalizing decisions of rejection or approval, a report is issued.

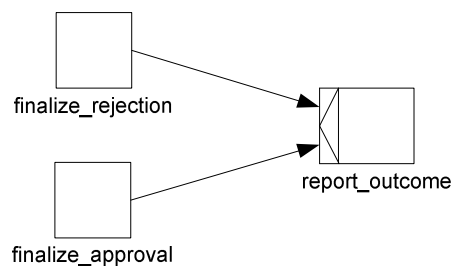


Figure 26 Simple Merge in YAWL.

Pattern 6 (Multi-choice)

Description Out of several branches, a number of branches are chosen based on user input.

Example

After the execution of activity *determine_teaching_evaluation*, execution of activity *organize_student_evaluation* may commence as well as execution of activity *organize_peer_review*. At least one of these two activities is executed, possible both.

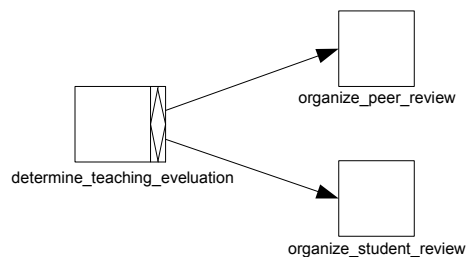


Figure 27 Multi-choice in YAWL

Pattern 7 (Synchronizing Merge)

Description A form of synchronisation where execution can proceed if and only if one of the incoming branches has completed and from the current state of the workflow it is not possible to reach a state where any of the other branches has completed.

Example

Consider again the example presented in Pattern 6 (Multi-choice). After activities *organize_student_evaluation* and *organization_peer_review* have finished, activity *interpret_results* could be scheduled. This activity should only await completion of those activities that were actually executed and itself be performed once.

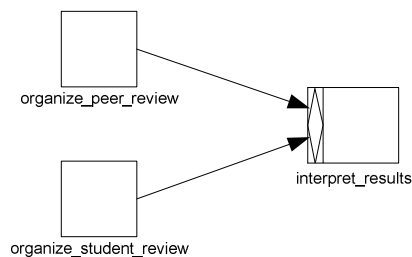


Figure 28 Synchronizing Merge in YAWL.

Pattern 8 (Multi- Merge)

Description It will execute the activity/task involved as many times as its incoming branches signal completion. Each enablement of an incoming branch results in the thread of control being passed to the subsequent branch.

Example

The *lay_foundations*, *order_materials* and *book_labourer* activities occur in parallel as a separate process branches. After each of them completes the *quality_review* activity is executed for each time one of the three tasks completes.

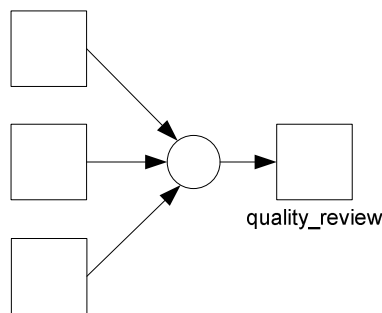


Figure 29 Multi-Merge in YAWL.

Pattern 9 (Discriminator)

Description It provides a form of synchronization for an activity where out of a number of incoming branches executing in parallel, the first branch to complete initiates the activity. When the other branches complete they do not cause another invocation of the activity.

Example

When handling a cardiac arrest, the *check_breathing* and *check_pulse* tasks run in parallel. Once the first of these has completed, the *triage* task is commenced. Completion of the other task is ignored and does not result in a second instance of the *triage* task.

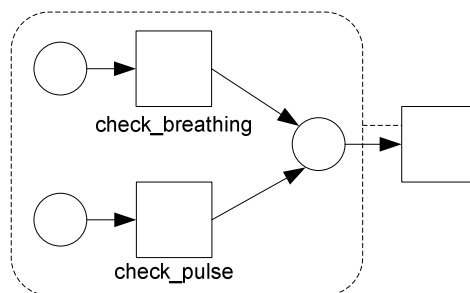


Figure 30 Discriminator in YAWL.

Description of Process Templates in YAWL

Never wait, executed every time (Pattern 8)

Consider booking of a business trip as an example, it starts with an OR-split *register* which enables task *booking_flight*, *reserving_hotel* and/or *renting_car*, activity *pay* is executed for each time one of the three tasks (i.e., *booking_flight*, *reserving_hotel* and *renting_car*)

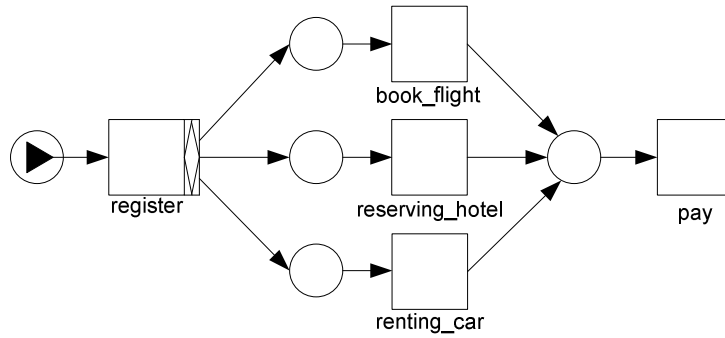


Figure 31 Task *pay* is executed each time one of the three preceding task completes.

Wait for all to come (Pattern 7)

Figure 4.15 is similar but combines the individual payments into one payment. Therefore, it waits until each of the tasks enabled by *register* completes. If only a flight is booked, there is no synchronisation. However, if the trip contains two or even three elements, task *pay* is delayed until all have completed.

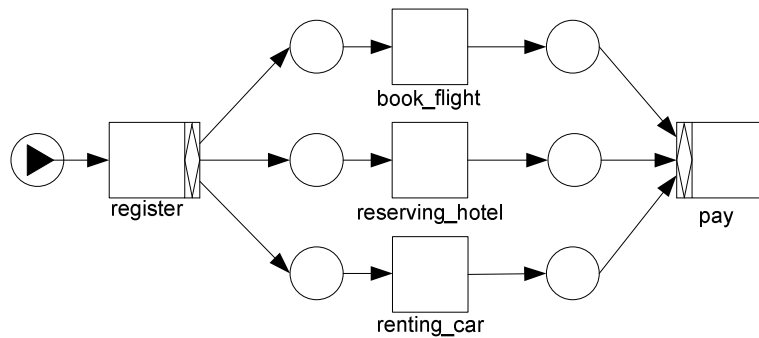


Figure 32 Task *pay* is executed only once, i.e., after all started tasks have completed.

Wait for first to come and ignore others (Pattern 9)

Figure 4.16 enables all three activities *booking_flight*, *reserving_hotel* and *renting_car*, activity *pay* is executed after the first task is completed. After the payment all running tasks are cancelled.

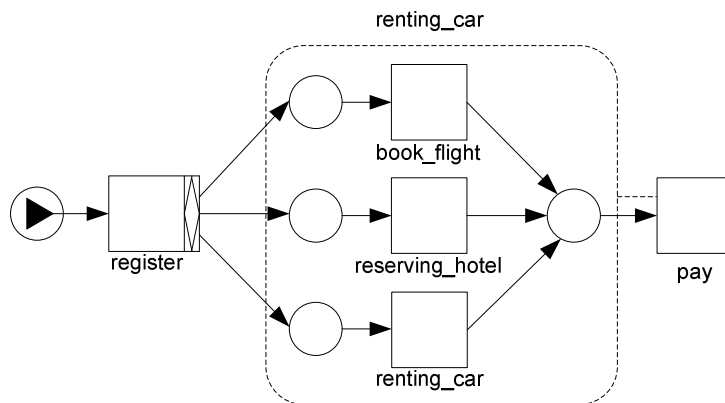


Figure 33 Task pay is executed only once, i.e., after the first task has completed.