A view based analysis of workflow modeling languages

Martin Vasko and Schahram Dustdar
Vienna University of Technology, Distributed Systems Group (DSG),
Information Systems Institute, A-1040 Wien, Argentinierstrasse 8/184-1, Austria,
e0025379@student.tuwien.ac.at | dustdar@infosys.tuwien.ac.at

Abstract

The different approaches of emerging workflow modeling languages are manifold. Today, there exist many notations for workflow modeling with various specializations on different domains. In this paper we analyze three well known business process (workflow) modeling notations for their support for elaborated key aspects in workflow modeling. The aim of this paper is to discuss their differences and commonalities concerning these aspects.

1. Introduction

This paper discusses the commonalities and the differences of well known workflow modeling languages: BPEL, BPMN and YAWL. After a short introduction of the common features of each language, we are going to skip right into the strengths and weaknesses of them. These three modeling languages cover a broad spectrum of workflow design notations. BPEL arose from a consortium of industry leading companies and therefore has strong vendor - background and offers many implementations. YAWL on the other side resulted from a formal specification of workflows [1]. This scientific approach aims at a more structured approach at the formal specification of different kinds of dependencies in workflow languages. The control – flow perspective has been taken into focus but YAWL is not only limited to this perspective. Current research is targeting data and resource patterns. BPMN addresses the lack of B2B standard notations. The visual notation will give organizations the possibility to elaborate and optimize their inherent business processes and communicate them to partners to simplify business – to – business transactions.

The motivation for business process modeling notations lies in the possibility to formalize complex orchestrations of formerly isolated Web Services. Collections of services are assembled to a business process.

Our comparison takes the aspects defined by Jablonski and Bussler[2] as a basis and analyzes the support in the three notations. First we analyze the functional aspect in the three languages. Secondly we discuss the support for the workflow patterns elaborated by van der Aalst et al. [1] as part of the behavioral aspect. The informational aspect is mostly defined by data and data flow and defines the third aspect. BPEL, BPMN and YAWL are examined for their support for this aspect.

2. Modeling languages

In this section, we give a short overview about the used standards of modeling languages. The structure and commonly used samples are being introduced.

2.1. BPEL4WS

BPEL4WS (Business Process Execution Language for Web Services) is currently standardized by OASIS lead by a consortium of industry heavyweights such as Microsoft, IBM, SAP, Siebel, and Oracle.

The goal of the Web Services effort is to achieve universal interoperation and loosely coupled systems. Layered on several XML specifications as WSDL 1.1, XML Schema 1.0 and XPath 1.0, BPEL is a Business Execution Language to enrich and standardize the commonly used protocols, especially for long–running business processes.

Among these XML based standards, WSDL has the most influence on BPEL. The whole process model of BPEL is layered on top of the service model defined by WSDL 1.1. The peer-to-peer interaction between the services described in WSDL is the conceptual basis for process interaction in BPEL.

The definition of such *business protocols* is involving a mutual visible message exchange behavior of each of the involved clients. The abstraction of BPEL hides any implementation details from the participating parties. The motivation of this design is obvious: Businesses do not want to share internal definitions and practices to others, maybe competitors. On the other hand, abstraction simplifies modular exchange and makes the whole system more flexible.

BPEL defines message-properties to identify protocol relevant data embedded in messages. This data can be used in a transparent or, alternatively, in an opaque way. Transparent data affects the public business protocol in a direct way, whereas opaque data is significant primarily for back-end systems.

In general BPEL can be applied in two ways: The business protocol can define abstract processes, which approaches data handling in a way that reflects the requirements of the workflow to execute.

Executable business processes on the other side determine the nature and sequence of Web Service interactions.

This basic concept model of distinction between abstract and executable processes makes it possible to export and import public aspects embodied in business protocols as processes or role templates.

2.2 BPMN

The primary goal of BPMN is to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the process, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor these processes. Thus, BPMN closes the gap between process design and process implementation.

Another design goal of BPMN is its compatibility to XML – based workflow languages like BPEL. Especially the visualization of processes designed with these notations is important.

The intent of BPMN is to standardize the business process design notation. Because of the complex domain of business process design, BPMN covers a variety of different modeling techniques and allows the creation of end-to-end business processes. The structural elements of BPMN will allow the viewer to easy differentiate between diverse sections of the diagram. The basic structure types of processes are the following:

2.2.1 Private (Internal) business process
Private business processes are those internal to a specific organization and concern the workflow definitions in general.

2.2.2 Abstract (Public) business process
Abstract business processes define the interaction between a private business process and other processes or participants.

2.2.3 Collaboration (Global) business process
Collaboration business processes describe the interactions between two or more business entities.

Currently these types of processes have not been standardized and so the definitions are in a flux. The World Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS) are addressing these terms.

For this work, the mapping of BPMN to other languages is of interest. The notation covers such a wide range of usage that it will map to more than one lower – level modeling language. As said above, BPEL is the primary language that BPMN will map to. Indeed, the mapping is restricted to a single executable private business process. Abstract processes will be mapped to Web Service Interface specifications, comparable to BPEL. The Collaboration model of BPMN may be mapped to Collaboration models such as ebXML BPSS, RosettaNet and the W3C Choreography Working group specification.

2.3 YAWL

To give a short overview about YAWL, we will introduce the language and its core design. The details are going to be analyzed in the following section, considering workflow patterns. YAWL is inspired by high-level Petri nets, but extended by some additional features to meet the requirements, defined by the workflow patterns elaborated by van der Aalst et. al. [1, 9].

Especially the models defining multiple instances, advanced synchronization and cancellation, are beyond the scope of high level Petri nets.

A definition of a workflow specification in YAWL is a set of extended workflow nets (EWF-nets) which have a tree-like structure. There are two types of tasks (the authors use the term task rather than activity to remain consistent to earlier work on workflow systems). Composite tasks consist of other composite or atomic tasks and refer to a EWF-net. Atomic tasks are no composition of tasks and do not have any subdefinitions. In the tree structure of EWF-nets the so called top level workflow does not map composite tasks. Conditions can be interpreted as places and each EWF-net has one unique input and one output condition.

Each task can have multiple instances, although the maximum and minimum number of instances can be configured.

Currently YAWL is implemented in a system consisting of a YAWL Designer and a YAWL engine, realized as a server side web-module. In the current release (Beta 3), the module runs on the Apache Tomcat Server. The engine verifies and registers the tasks, which will be stored in the YAWL repository. This repository manages a collection of executable workflow specifications. After successful deployment, the workflows can be invoked.

The environment of the YAWL system consists of so called YAWL services, explained below.

♦ YAWL worklist handler

This component assigns work to the users. Through this service, users can accept work items and signal their completion.

♦ YAWL webservice broker

The webservice broker acts as the interface between the YAWL engine and other webservices.

♦ YAWL interop broker

The interoperability broker provides the possibility to interconnect different workflow engines.

♦ Custom YAWL service

A custom YAWL service can be of any kind, the figure illustrates the modularity of the services. This mechanism makes it possible for external services to communicate through a “gateway” with the YAWL engine.

After this short introduction of the three workflow modeling languages we are going to analyze, we give a short overview about those aspects we will discuss.

2.4 Aspects

Aspects are a collection of key functionalities of different knowledge domains of a workflow system. Jablonski et.al.[2] identified five key aspects which we will concentrate on to broaden our focus. To provide comprehensive support for workflow modeling we try to elaborate identifying patterns of every key aspect. The three notations previously introduced are compared for the realization for these patterns. First we will take a look at the functional aspect.

2.4.1 Functional Aspect. The functional aspect concentrates on what has to be done in a workflow process. It defines the hierarchical context of tasks and sub-tasks. Petkov et.al. [4] define two modeling concepts of workflows. *Compositional* workflows either contain sub-workflows and are called composite workflows or are of atomic kind. To analyze the support for this functional aspect by the three notations we compare the possibility for nesting constructs.

Constraints of workflows are a set of rules covering consistency. There are three kinds of constraints mentioned: Enter constraints are being evaluated before the initiation of a workflow process. Runtime constraints are verified during execution of the process and exit constraints are checked at the completion time of a workflow.

2.4.2 Behavioral Aspect. This aspect concentrates on the control flow of the workflow model in the whole. Execution order of the different activities is of main interest. Van der Aalst et.al.[1] provide metrics for this aspect. They elaborated a collection of patterns which represent the smallest common set of activities to design a complex workflow. We take these patterns as our starting point for comparing the support by BPEL, BPMN and YAWL.

The workflow patterns are presented as a benchmark for the functionality of workflow systems in a number of publications. We elaborate the commonalities and differences between the three notations on the basis of an abstract of the patterns. The interested reader may refer to [1] for further details.

2.4.3 Informational Aspect. Jablonski and Bussler [2] divide the Informational Aspect into a *data* and a *data flow* perspective. The main focus lies on the data and to provide relevant data at the right time. Data itself is divided into two different kinds: *Production data* and *Control data*.

Production data is consumed and produced by the workflow system, whereas control data defines the management of this process. This definition is closely connected to the workflow control data definitions by the Workflow Management Coalition [5].

To bring this aspect in our analysis, we define a metric for the three notations covering this aspect. The languages should support the data type they are operating with. Russel et al. [5] defined workflow data patterns, which we reference in general and analyze the notations for the support for these patterns.

After this short overview of the aspects we examine the commonalities more closely.

3. Comparison

First we survey the functional aspect and the nesting constructs of BPMN, BPEL and YAWL.

3.1 Functional Aspect

As we want to define *what* has to be done in the workflow we keep focus on the meta modeling capabilities of BPMN. According to the specification (actual in revision 1.0 [7]) a core element named the **Data Object** is provided to hold information about what has to be done in associated tasks. Its visual representation is comparable to UML Notes. As an enhancement of Data Objects acts a textual annotation.

Richer support is provided for compositions in BPMN. Core Business Process Diagrams contain **Sub-Processes**. As the name already says, these processes represent a visual approach for "hiding" workflow complexity from the modeler. Collapsed Sub-processes conceal the internal definitions of a process. Expanded processes establish a view to the workflow internals. This feature allows a kind of abstraction in workflow modeling. In terms of the functional aspect the possibility for nested/embedded sub-processes is of great interest. This element is an implicit extension of the previously explained sub-process and explicitly shares the same set of data as its parent process. Data needs to be passed to the sub-process.

Since BPEL [8] is based on WSDL, the notation inherits the description features of WSDL which fit perfect to the functional aspect. BPEL's process model is layered on top of the service model provided by WSDL. As part of the notation, the most relevant element of WSDL concerning the functional aspect is the **operation** attribute. It provides an abstract description of an action supported by the service.

YAWL [9] extends workflow nets [10] among other things by composite tasks. These tasks refer to one or more EWF – nets which again consist of one or more tasks or composite tasks. These constructs provide good nesting features for modeling. On the other hand we found no support for any kind of meta-modeling. In other words it is not possible to describe *what* will be done by a concrete workflow definition in YAWL.

3.2 Behavioral Aspect

As an in depth analysis of all provided patterns for all three notations would go beyond the scope of this article we summarized the patterns into six different groups. A more detailed discussion will be provided in our future work.

Commencing the basic control patterns, all three notations provide support for sequential execution, conditional and simple decision based branching.

The Advanced Branching and Synchronization patterns provide some difficulties. BPMN solves many of these patterns by the usage of gateways. Gateways are used to control branching and merging flows. If there are multiple paths (incoming and outgoing) involved, these core elements of BPMN are extended by Tokens which take control over these multiple possibilities. Only the Multiple Merge pattern solution does not use gateways because the number of Tokens involved is not controlled. For each incoming Sequence Flow a separate instance of activity will be generated.

BPEL on the other hand does not provide rich support for this group of patterns. Only the Multiple Choice pattern can be realized by using the `flow` statement of BPEL. In listing 1 an example for the syntax of this statement is provided.

Listing 1

```
<flow>
    <links>
        <link name="L1"/>
        <link name="L2"/>
        <link name="L1s"/>
        <link name="L2s"/>
    </links>
    <empty>
        <source                    linkName="L1"
transitionCondition="C1" />
        <source                    linkName="L2"
transitionCondition="C2" />
    </empty>
    activityA1
        <target linkName="L1">
        <source linkName="L1s">
    activityA2
        <target linkName="L2">
        <source linkName="L2s">
</flow>
```

The underlying principle of this structure is called *dead-path elimination* and states, that the value of an incoming link is propagated to its outgoing link. According to listing 1, a multiple choice pattern occurs, if the conditions C1 and C2 evaluate to true. If this is the case, activity A1 and activity A2 are executed.

The numerous split - and join – tasks of YAWL provide very good support for these patterns. In Figure 1, task B, C, D or any combination thereof is executed after task A. The tasks are connected via an OR-split.

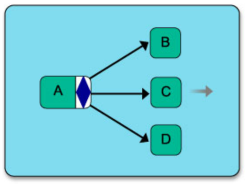

Figure 1. YAWL: Multiple Choice

Therefore, it is a solution for a Multiple Choice pattern and illustrates the visual representation of YAWL. On the other hand a synchronizing merge pattern is realized by the OR – join in YAWL. This element is presented in figure 2. The tasks B, C and D in Figure 2 are ORed and task A is executed if all of them are completed. The join stalls execution until all participating tasks are finished.

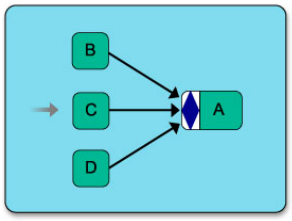

Figure 2. YAWL: Synchronizing Merge

Structural patterns deal with loops and the independence of separate process paths. BPMN provides upstream activities which make an access of the process in an arbitrary way possible. Concerning the Implicit Termination of a process, BPMN provides End Events. These Events have internal markers that indicate with a result the ending of the current path. This Terminate End Event causes the entire path to be stopped, even if there are activities that have not been started before or are still active.

Although BPEL provides a `while` statement, it is not supported to jump into the process in an arbitrary way, and as a consequence, the Arbitrary Cycle pattern is not supported. Realizing the explicit termination, the `flow` statement ends when all structured activities defined in the scope of the statement completed. Using link activities, a subprocess can have multiple tasks not being source of any link without requiring a unique termination action.

Since YAWL defines no restrictions on cycles the Arbitrary Cycle pattern is supported. YAWL provides no solution for the Implicit Termination pattern. This design decision was made to force the modeler of the workflow process to think carefully for the termination construct of the process. It is possible to achieve similar behavior through the connection of all final tasks with an or-join.

The next group of patterns we are going to analyze concern multiple instances. Problems occurring in the scope of the activities involved are synchronization and knowledge of the number of instances to be created.

BPMN provides no possibility to spawn multiple instances in an unsynchronized way. This absence makes the realization of the pattern dealing with multiple instances without synchronization impossible. Furthermore, only a workaround is possible when the number of required instances is not known before runtime. This pattern is comparable to the functional behavior of a *while* loop, unless the execution takes place in parallel.

In all other patterns the number of required instances is known at runtime or at design time. These patterns are supported by BPMN by the definition of powerful attributes in the activity element covering multiple instances control.

BPEL provides rich support for patterns attending multiple instances. The *flow* statement for example enables concurrency and synchronization within a set of activities by enclosing a group of statements. If there is no synchronization required, the *receive* statement provides an additional *createInstance* attribute. If this is set to *yes* for every incoming link an instance is spawned. For patterns without runtime knowledge (neither at design time, nor at runtime the number of needed instances is known) BPEL provides the *pick* statement. `Pick` enables a messaging-mechanism (comparable to exceptions in common programming languages) embedded in a `while` statement. In listing 2 there are three different message types defined: `StartNewActivityA` to indicate that a new instance is required. According to the previous pattern, this message can be created at runtime. `ActivityAFinished` signals, that the execution of an instance has come to an end. `NoMoreInstances` represents the end of the creation. Note that this code excerpt is only a workaround for the two patterns and BPEL does not support these patterns directly.

Listing 2

```
moreInstances:=True
i:=0
<while moreInstances OR i>0>
    <pick>
        <onMessage StartNewActivityA>
            invoke activityA
            i:=i+1
        </onMessage>
        <onMessage ActivityAFinished>
            i:=i-1
        </onMessage>
        <onMessage NoMoreInstances>
            moreInstances:=False
        </onMessage>
    <pick>
</while>
```

The implementation and visual notation support for patterns dealing with multiple instances in YAWL is straight forward as explained in Figure 3. After activity A has completed n tasks of B are created. When all instances of B have come to an end, the final task C is executed. If the number of instances is known at design time, N has to be defined at design time. Assumed that the number of required instances is known at runtime, the number of n is defined by XQuery expressions which are evaluated at runtime. In the case of no knowledge about the number of required instances, XQueries are evaluated to determine the n extended by the possibility to add instances dynamically.

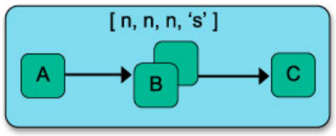

Figure 3. YAWL: Multiple instances with a Priori Design Time knowledge

The next group of patterns defines how the behavior of a business process may sometimes be affected by outside tasks, not under control of the workflow system. BPMN defines rich messaging capabilities and provides good support for these patterns with the Exclusive Data Based Gateway.

This structure represents a point in the workflow where one of several alternative branches is chosen. The decision is based, as required, on an external Event and is followed by Intermediate Events.

BPMN features the possibility to visualize ad-hoc processes. This type of process represents a set of activities that can be executed in an arbitrary order to support interleaved parallel routing. Although BPMN provides these rich messaging functionalities, the realization of the Milestone pattern is only possible through a workaround. Here we used the inter-process messaging provided to communicate between two defined sub-processes.

BPEL provides comparable structures with the previous mentioned *pick* statement. The specification of BPEL defines `scopes`. `Scopes` can be serialized by setting the `variableAccessSerializable` attribute to `"yes"`. This attribute guarantees concurrency control in governing access to shared variables. Serialized scopes must not be nested. This is essential for the arbitrary access of the execution of different activities as in the pattern interleaved parallel routing pattern defined. As in BPMN, BPEL does not provide direct support for the Milestone pattern. It is possible to achieve similar functionality with the messaging mechanism of `pick`.

In all State-based patterns YAWL makes use of the mutual exclusion mechanism inherited from Petri Nets. Task C in Figure 4 can only be executed after task A has finished.

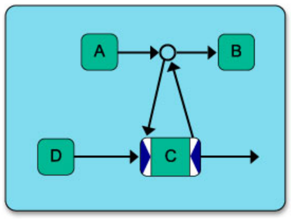

Figure 4. YAWL: Milestone pattern

A special requirement is achieved by positioning the mutual exclusion before task B. So the commitment of task C is delimited to execute before task B.

3.3 Informational Aspect

The integral part of this aspect is data which is being processed in workflow systems. Russell et al. [5] divided data by the following characteristics: *Data visibility*, *Data interaction*, *Data transfer* and *Data-based routing*.

We are going to compare the three notations by their support for the four characteristics. The interested reader may refer to [5] for detailed workflow data patterns.

Data visibility concerns in general the support for access constraints of data elements in different tasks, scopes or in case of multiple instances distinct instances.

In BPMN data visibility is mainly implemented by so called *Pools* and *Lanes*. A pool provides the possibility to partition a set of activities from other pools. To achieve a splitting within the pools, lanes are provided. These elements deal with enfolding parts of workflow processes. Beside a more detailed view BPMN provides sub-processes. These elements hide in the collapsed state all internal process definitions and processed data from other tasks. The data interchange takes place over the powerful messaging mechanism of BPMN, as described in the next section.

In BPEL the data visibility is controlled by the definition of *scopes*. A scope can provide fault handlers, event handler, a compensation handler, data variables and correlation sets. From the view of data visibility the *variableAccessSerializable* attribute of a scope is very interesting. If this attribute is set to *yes* the scope provides concurrency control in governing access to shared variables. The second relevant features of scopes in BPEL are correlation sets. All values of the properties of a correlation set must be identical for all messages and invocations that carry this initiated set and are executed in the current scope until its completion. This guarantees data integrity during the performance of the scope.

YAWL provides rich support concerning data visibility. The power of YAWL lies in the extensive use of XQuery to map data between different levels of abstraction in the use of multiple instances. Especially the treat of data in association with multiple instances is far from trivial. All required data needs to be split over all spawned instances and after the tasks all completed, the data must be aggregated again to represent the result of all instances.

As data interaction and data transfer are tightly coupled, we are going to summarize these two aspects in one paragraph.

BPMN implements a rich messaging mechanism which deals with data transfer. The previous mentioned pools support message flow. All other elements described earlier support this type of data interaction too. The data interaction between defined elements within the workflow is realized well in BPMN. On the other side the data interaction with external sources, operated outside the context of the workflow engine is not provided. The modeler is able to define some kind of workaround by the use of pools for external sources.

In BPEL the transfer of data is defined in the *assign* statement. This statement copies data from one variable to another, as well as construct and insert new data using expressions. The motivation for the support for expressions arose from the need for simple computation, such as incrementing sequence numbers. The to- and from-specs defined in an assignment have to be from the same type (for example the from-spec is a variable of a WSDL message type and the to-spec is a variable of a WSDL message type). This loose definition makes interaction with external data sources easier than in BPMN.

In YAWL the use of XQuery expressions is mostly used by the decomposition of different tasks and to access nodes in the workflow definition in an arbitrary way. This leads us to the last data aspect view, data based routing.

As mentioned previously XQuery is used in YAWL to retrieve and interpret information from different sources used during the workflow. Data-based routing is provided in YAWL through this powerful feature, in conjunction with XPath. These two XML – languages together form a good basis for numerous data based routing mechanisms.

BPMN uses different kinds of gateways to enable routing in the process definition. These gateways generally provide OR/XOR/AND and more complex decision based functions. Important for this aspect is the fact, that these elements of BPMN also accept data as a decision basis. That means that data-based routing is made possible by definition, and no additional features, such as XPath or XQuery in YAWL is needed to support this part of aspect.

BPEL provides through the utilization of link conditions and the exception mechanism support for any kind of task pre- and post conditions. Only a data-based task trigger can be realized indirectly.

4. Conclusion

The three analyzed business modeling notations all have their strengths and weaknesses. BPMN is a well elaborated visual modeling notation which provides good support for behavioral aspects of workflow design and is able to map to BPEL4WS. Although it is not extensible to define organizational structures, functional breakdowns, data and informational models and business rules it provides a de-facto standard to model business processes. BPMN is targeted to all business users and tries to close the gap between business process design and business process implementation.

BPEL is probably the most frequently used and widely accepted industry business process execution language. It is based on Web Service standards and supports most of the elaborated workflow patterns. The notation has its weaknesses in dealing with data structures and complex control flows. As the process model is based on WSDL it perfectly fits in Web Service architectures.

YAWL extends Petri nets by numerous mechanisms and features to enable complex multiple instances workflows and advanced branching and synchronization patterns. BPEL is based on well known web service technologies. Its strengths lie in the good support for workflow patterns and its powerful data structures support by extensive use of XQuery. Regardless it has to penetrate the web service market to unleash its full potential. The reference implementation of the YAWL system provides a good starting point.

Table 1. Comparison of notations[1]

	BPEL	BPMN	YAWL
Functional Aspect			
Compositions	X	X	X
Constraints	X	X	X
Behavioral Aspect			
Basic control patterns	X	X	X
Advanced branching and synchronization patterns	-	X	X
Structural patterns	O	X	X*
Patterns involving multiple instances	X	O	X
State – based patterns	X*	X*	X
Cancelation patterns	X	X	X
Informational Aspect			
Seperation of Control/Production Flow	O	O	-*

Based on the previously introduced aspects we want to give a short outlook for future work in Business process modeling notations. Will van der Aalst et al. [1] provide a comprehensive reference for the behavioral aspect a notation has to be compliant with. They already published data and resource patterns and are currently working on an implementation for YAWL to demonstrate the support for these patterns [12,13]. Beside YAWL BPEL4WS has to support these patterns representing a reference model for informational and operational aspects. Apart from these scientific interests the analyzed workflow modeling notations and their implementations should be widened in usage and expanded in numerous features like modeling tools for domain and problem specific scenarios to simplify their insert.

5. References

[1] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski and A.P. Barros. *Workflow Patterns*, Distributed and Parallel Databases, 14, 5-51, 2003, Kluwer.

[2] S. Jablonski and C. Bussler. Workflow Management: Modeling Concepts, Architecture and Implementation. International Thomson Computer Press, 1996.

[3] W.M.P. van der Aalst, L. Aldred, M. Dumas and A.H.M ter Hofstede, *Design and Implementation of the YAWL System*, Proceedings of The 16th International Conference on Advanced Information Systems Engineering, 2004, Springer Verlag.

[4] Simeon Petkov, Eyal Oren and Armin Haller. *Aspects in Workflow Management,* DERI Technical Report, April 2005.

[5] Nick Russel, A. H. M. ter Hofstede, David Edmond, *Workflow Data Patterns*

[6] C. Brussler, *Organisationsverwaltung in Workflow-Management-System.* Ph.D. thesis, University of Erlangen 1997.

[7] BPMN Specification, Version 1.0 from May 3, 2004; http://www.bpmn.org, available on Tuesday, 24. May, 2005

[8] BPEL4WS Specification, Version 1.1 from May, 2003; http://www-128.ibm.com/developerworks/library/specification/ws-bpel/ available on Tuesday, 24. May, 2005

[9] W.M.P. van der Aalst and A.H.M. ter Hofstede, *YAWL: Yet Another Workflow Language,* Information Systems, 30(4):245-275, 2005.

[10] W.M.P. van der Aalst, *The application of Petri Nets to Workflow Management,* The Journal of Circuits, Systems and Computers, 8(1):21-66, 1998.

[11] W.M.P. van der Aalst et. al., *Design and Implementation of the YAWL system*, Proceedings of The 16th International Conference on Advanced Information Systems Engineering (CAiSE 04), Riga, Latvia, June 2004

[12] N. Russell, A.H.M. ter Hofstede, D. Edmond, W.M.P van der Aalst, *Workflow Resource Patterns*, BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven, The Netherlands, 2004.

[13] N. Russell, A.H.M. ter Hofstede, D. Edmond, W.M.P van der Aalst, *Workflow Data Patterns,* Technical Report FIT-TR-2004-01, Queensland University of Technology, Brisbane, Australia, April 2004

[1] X … This feature is full supported,
O … a workaround is possible,
* … Supported with limitations.