A Notation Evaluation of BPMN and UML Activity Diagrams

Lauri Eloranta
Eero Kallio
Ilkka Terho

T-86.5161 Special Course in Information Systems Integration (2006)

Table of contents

Abstract

In this study we evaluate the representation power of two business process modeling notations: business process modeling notation (BPMN) version 1.1 (OMG 2006a) and unified modeling language activity diagrams (UML AD) version 2.0 (OMG 2006b). By representation power we mean the notations ability to represent real world phenomena. This means, that in the heart of this study is the question: do these notations have any differences in their representation power? I.e. should there be some real world phenomena that either of these business process notations is not able to model?

Our general research viewpoint comes from two analysis frameworks that are used to analyze modeling notations and their representation power: the *Bunge-Weber-Wand-model (BWW)* and the *Workflow patterns framework* (Workflow Patterns 2006). The BWW-model is a reference model that can be used to find out how well a given notation can model real world phenomena generalized by the BWW-model. The workflow patterns framework on the other hand gives a general set of business process patterns that can be evaluated against a notation to find out, if there are some patterns that a notation cannot model.

The findings of our study suggest that there are only a few differences in representation power between the notations. If one notation should be chosen to be better in representation power it would be PBMN, but the difference is marginal.

Keywords: Business Process Modeling, Business Process Modeling Notation, Unified Modeling Language Activity Diagram, Workflow Patterns framework, BWW-model, representation power.

1 INTRODUCTION

This research is done as a work for the Helsinki University of Technology course T-86.5161 Special Course in Information Systems Integration (2006): Business Process Management. The research group has an interest to investigate business process modeling notations[1] to be used in working life now and in the future.

The aim of our research is to compare two different ways to model a business process: business process modeling notation (BPMN) version 1.1 (OMG 2006a) and unified modeling language activity diagrams (UML AD) version 2.0 (OMG 2006b). Our evaluation of the two modeling notations concentrates solely on the given specifications[2] of these notations.

Our comparison is based on the representation power of the notations, specified by the specification of the notation. By representation power we mean the notation's ability to represent real world phenomena. This means, that in the hearth of this study is the question: do these notations have any differences in their representation power? I.e. are there some real world phenomena that either of these business process notations is not able to model?

Out of the scope of this research are matters that do not directly relate to the given notations. In our evaluation we do not consider, for example, matters like tool support for the notations or how well these notations and process models made with these notations, can be mapped to executable languages or how well a modeled process can be executed. Though, this does not mean, that these matters are not worth considering in further research.

Our general research viewpoint comes from two analysis frameworks that are used to analyze modeling notations: the *Bunge-Weber-Wand-model (BWW)* and the *Workflow patterns framework* (Workflow Patterns 2006). The BWW-model is a reference model that can be used to find out how well a given notation can model real world phenomena generalized by the BWW-model (Rosemann and Green 2002; Recker et al. 2005). The workflow patterns framework on the other hand gives a general set of business process patterns that can be evaluated against a notation to find out, if there are some patterns that the notation cannot model (Russell et al. 2005a, 2005b, 2006).

Both of the frameworks are chosen for their ability to analyze the representation power of a given notation. They are also both well known and well cited frameworks that have been used successfully in many researches (Rosemann and Green 2002; Opdahl and Henderson-Sellers 2002; White 2004b; Russell et al. 2005a, 2005b, 2006).

[1] By *notation* we mean the visual appearance of the graphical elements and the semantics of the elements.
[2] By *specification* we mean the actual specification document of a given notation.

Thus, our *research problem* is:

- Are there any differences in the representation power between the given notations?

Further, *research questions* are as:

- Are there any differences between the given notations when they are analyzed with work flow patterns -framework?
- Are there any differences between the given notations when they are analyzed with BWW-model?

1.1 Research methods

The research is done based on a literature analysis. Our main sources of information are articles that handle matters like UML AD and BPMN in general, Workflow patterns framework and BWW-model. Also the specifications of these two notations are included in the analysis.

Our literature analysis is based on the methodology described by Webster and Watson (2002). We tried to follow the structured approach given in their paper and followed citations backward and forward as Webster and Watson (2002) suggested. Because our research subject is quite a new in the field of computer science, more weight was put on the academic searches.

We did an extensive literature study to gather knowledge about the topic. First we tried out three methods to search data. We tried out Google Scholar (scholar.google.fi), UMI Proquest (proquest.umi.com/login) and Nelli portal (www.nelliportaali.fi). It was obvious that concerning this topic Google Scholar is able to find much more results than the two other methods. We got hundreds of hits with it. Other methods were not able to retrieve as many hits.

We planned an initial set of key words that was the base for our search. We divided the key words between our group members. Table 1 presents the key words and hits that were found with Google Scholar. Everyone went through his hits and pick articles that best match our topic. The result sets for some key words were too big and some focusing words were used to be able to go through these hits.

Table 1

Key words	Hits
BPMN	473
"business process modeling notation"	242
UML "activity diagram"	3600
"unified modeling language" "activity diagram"	2200
BPMN UML	264
BPMN UML "activity diagram"	73
BWW model	1320

workflow pattern	15900

We selected 45 articles that we though best match our topic. We divided the papers between three persons and everyone read through his own set of papers. Based on this study we found twenty three articles that were useful for our paper. We have used about twenty of those articles and in addition we have taken some papers from outside the literature study which we did not find with our original key words but which were found when doing this study.

1.2 Structure of the report

This report is divided into three parts:

1. In the first part we introduce the basic concepts of this report: The BPMN and UML AD notations and the scope[3] of these specifications. We also describe the theoretical backgrounds of the used Workflow Pattern framework and Bunge-Weber-Wand model in chapters four and five.

2. In the second part we examine the differences in the representation power of the notations. Thus, in chapters six and seven we sum up the findings of the earlier studies considering these matters.

3. In the third part we make concluding remarks (chapter eight) and vision further directions for research concerning this matter (chapter ten).

[3] By *scope* we mean the planned scope of what for the notation is suitable, according to the specification.

2 Introduction to BPMN

In this chapter we describe the Business Process Modeling Notation (BPMN) 1.1 and for what purpose it is used, according to the specification.

2.1 The Basics of BPMN

The Business Process Management Initiative (BPMI.org) consisting of software vendors proposed the Business Process Modeling Notation (BPMN) 1.0 specification in May, 2004. The Object Management Group (OMG) adopted it for standardization purposes in February, 2006. Currently version 1.1 of the notation is in the finalization phase. At the moment there are at least 30 vendors supporting modeling with BPMN. (Recker et al. 2006.)

The primary goal of the new notation is to make business process modeling easier and reduce the gap between technical and business people. Several other notations and methodologies were reviewed to gather the best ideas to a single notation. These include e.g. UML, IDEF, ebXML, RosettaNet, LOVeM and Event Process Chains (OMG 2006a). The BPMN is still in an evolving phase and it should be developed further. Ultimately there should be a notation that supports both graphical and formal presentation of the business process. (OMG 2006a.)

2.2 The BPMN Notation

BPMN supports only those concepts that are needed when modeling business processes. Other business modeling is out of scope. BPMN is not applicable e.g. for modeling strategy or organizational structures. (OMG 2006a.)

The notation defines the Business Process Diagram (BPD) which is based on flowcharts. The Business Process Model combines the BPD and the flow controls which define the execution order of activities. (White 2004a.)

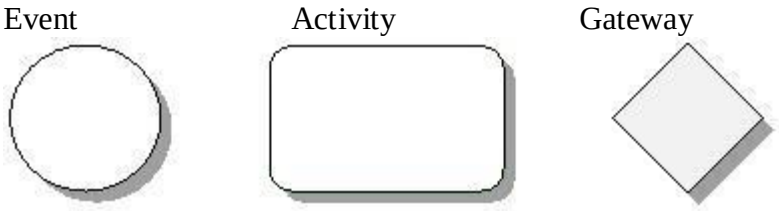
The basic idea of the notation is that there are only a few basic elements which are easy to learn. More advanced elements can be used to be able to model more complex processes. The BPD core element set consists of eight elements. Overall there are thirty-eight distinct language constructs. Extensions can be used as a part of the model, but they should not change the shapes of the basic BPD elements. (OMG 2006a.)

2.2.1 BPD Elements

To keep the notation simple there are only four basic categories of elements:

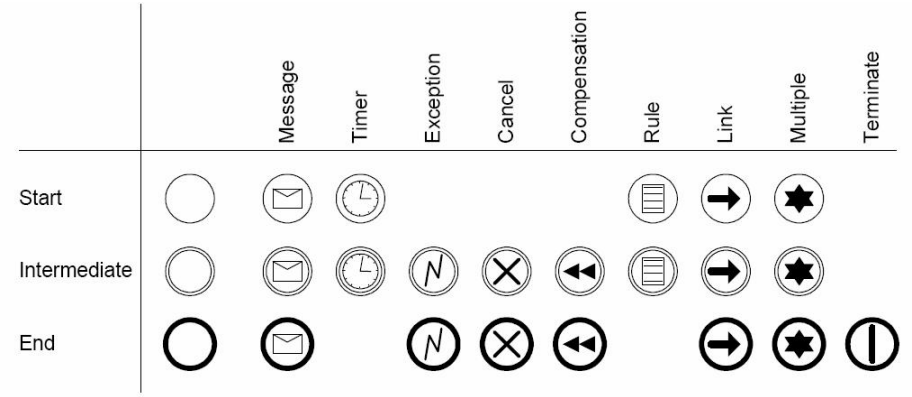
- Flow Objects
- Connecting Objects
- Swimlanes
- Artifacts

Flow Objects define the behavior of the business process.

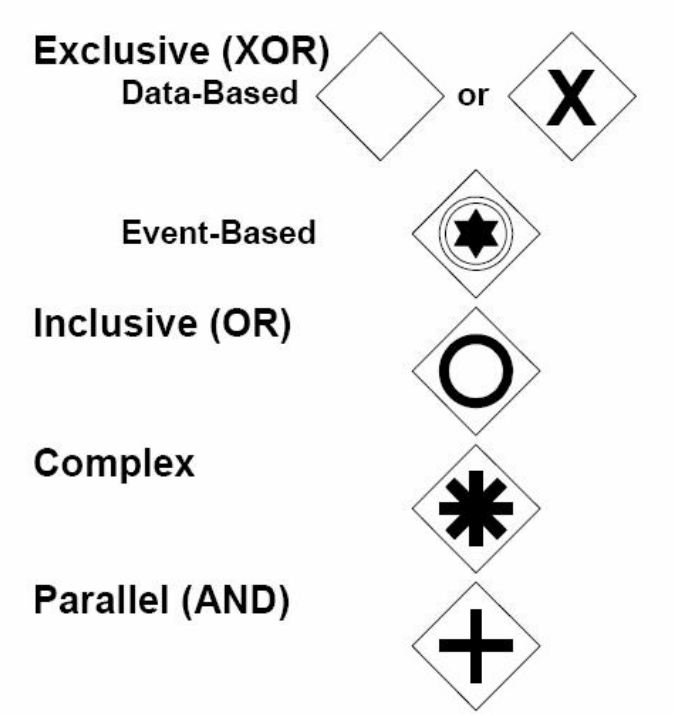


Flow Objects
Figure 1

Events happen during the business process. They usually have a trigger and a result. The type of an event defines when it affects the flow. An activity is something that the organization performs. An activity can be atomic or non-atomic. A gateway determines branching, forking, merging, and joining of paths. Internal Markers are used to define the behavior control.



Complete list of BPMN event types (Schnieders et al. 2004)
Figure 2



Complete list of BPMN gateway control types (OMG 2006a)
Figure 3

Connecting Objects connect flow objects to each other or to other information.

Connecting Objects
Figure 4

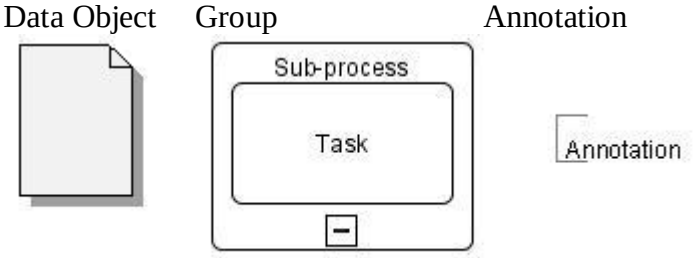
A Sequence Flow defines the execution order of activities. A Message Flow defines the flow of messages between business entities. An Association associates information to flow objects. Swim lanes group the modeling elements.

Lane
Figure 5

A Pool with two lanes
Figure 6

There can be only one participant in a Pool. Lanes are used to organize and categorize activities.

Artifacts



Artifacts
Figure 7

Artifacts do not have any direct impact on sequence or message flow. Data Objects represent what information is needed to perform some activity. A group element is used to group activities and it can be used for documenting or analyzing purposes. Annotation can be used to add additional text to the diagram.

2.3 Three levels of abstraction in BPMN

BPMN is designed to support end-to-end process modeling. To be able to model complex end-to-end processes different levels of abstraction can be used. The modeling can be started from high level activities and the data can be separated to several diagrams.

There are three basic levels of abstraction within an end-to-end BPMN model:

A Private (Internal) Business Process includes the activities that are internal to an organization. A Private process is the same as workflow. This level of abstraction represents details and private activities that should not be seen by other organizations. Interaction with external participants can also be included at this level.

An Abstract (Public) Business Process represents the activities that are needed for interaction with a private business process and another process or participant. Internal activities are not shown.

A Collaboration (Global) Business Process is a B2B level of abstraction. It represents the business process from the global point of view, not using any business entity's view. Only interactions between two or more business entities are shown. (OMG 2006a.)

2.4 Formalism and mapping in BPMN

The idea of BPMN is that it should be easy to use. That is why the BPMN notation is not formal. The specification of BPMN defines a mapping to Business Process Execution Language for Web Services (BPEL4WS) which is a formal mechanism for defining a business process (OMG 2006a). The execution of the process model is an important goal of BPMN. The automatic conversion between graphical and executable processes and the diagram exchange between conformant tools should be achieved to ensure the success of the notation. (Schnieders et al. 2004.)

3 Introduction to UML Activity Diagrams

In this chapter we describe the UML 2.0 Activity Diagrams and for what purpose they are used, according to the specification.

3.1 The Basics of UML

The Unified Modeling Language has been developed by the Object Management Group (OMG) for about ten years now and it is currently upgraded to version 2.0 (Wikipedia 2006). The scope of the UML language is very broad and it covers a large and diverse set of application domains. UML Activity Diagrams can be used to model a business process.

In UML 1.x an Activity Diagram is a variation of the UML State diagram where the states represent operations, and the transitions represent the activities that happen when the operation is complete (Wikipedia 2006). The UML 2.0 Activity Diagram (UML AD) was formed based on the ideas of Petri nets by using semantics defined in token flow (Wohed et al. 2005).

3.2 The UML Notation

UML AD is a behavior diagram whose fundamental unit is Action. An Action can have a set of inputs and outputs, and it may also change the state of the system. There are over 40 different types of actions, but the following are the most relevant for business process modeling: general Action concept, Action Event, Send Signal and Call Behavior Action constructs. (OMG 2006b.)

To illustrate the system's behavior the UML AD has elements called Activities. In diagrams these Activities are separated to edges and nodes. Edges are acting connectors between nodes in sequential order. Nodes can be Actions, Sub Activities, Data Objects or control nodes. (OMG 2006b.)

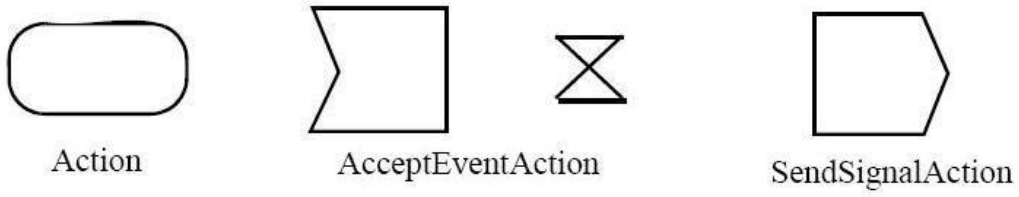
The basic idea of the UML AD is that there are elements that can be used for various purposes. To model business processes you have to know which elements fit best for you. When elements are fixed, the modeling is straightforward and easy to read for those who have the same kind of experience.

3.2.1 UML AD Elements

Activities can be separated into five main categories of elements:

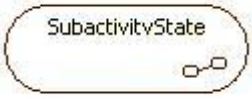
- Actions
- Sub Activities
- Data Objects
- Control nodes
- Partition

Actions define the behavior of the business process. They have inputs and outputs that might be also empty. Actions can change the state of the system.



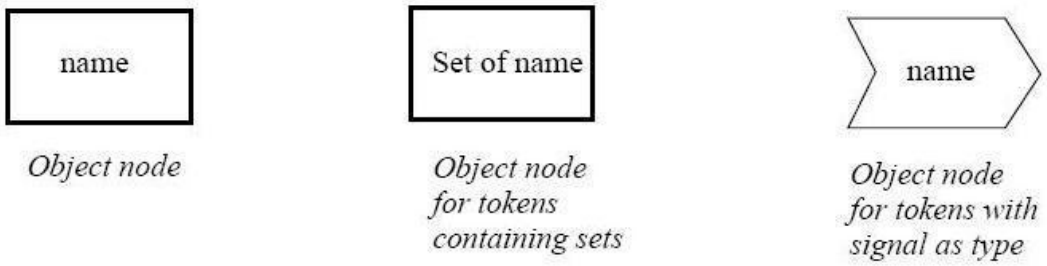
Actions (OMG 2006b)
Figure 8

Sub Activities hide the complexity of a model.



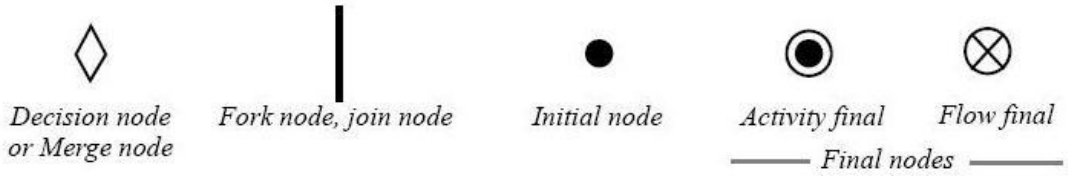
Sub Activities
Figure 9

A Data Object indicates an instance of a particular classifier, possibly in a particular state, that may be available at a particular point in the activity.



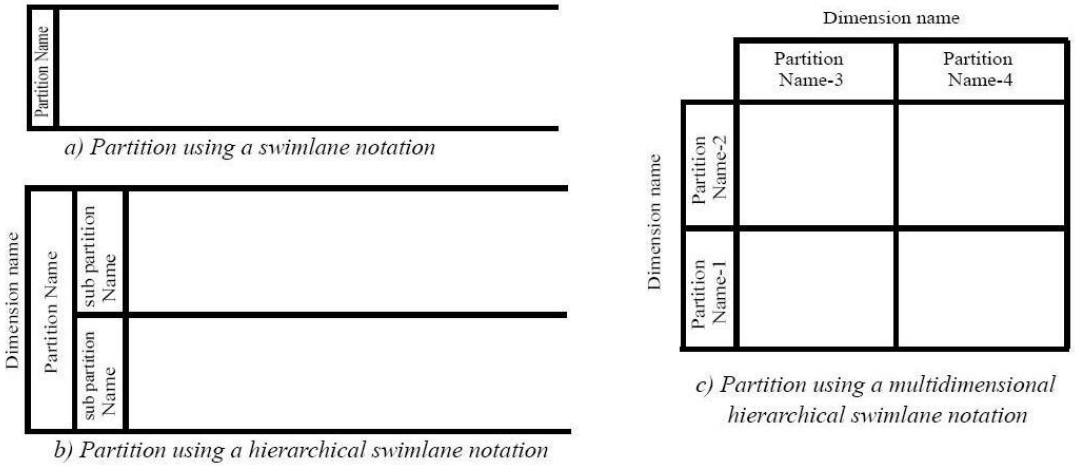
Data Objects (OMG 2006b)
Figure 10

A Control Node coordinates the flows between other nodes.



Control nodes (OMG 2006b)
Figure 11

Partitions divide the nodes and edges to constrain and show a view of the contained nodes. They often correspond to organizational units in a business model.



Partition (OMG 2006b)
Figure 12

3.3 Business Process Modeling with UML AD

The UML AD specification does not speak about business process modeling. It says that Activities may be applied to organizational modeling for business process engineering and workflow. (OMG 2006b.)

Swimlanes are used to express hierarchical partitioning of a process. There can be only one controller with one state per diagram which handles the execution of a process. There are no concepts for adding many process participants to the same diagram each with their own controller. (OMG 2006b.) This makes it difficult to model a collaboration process which is supported by BPMN.

3.4 Formalism and Mapping in UML

There are different compliance levels in UML. Every compliance level adds more capabilities over the former compliance level. The lowest compliance level is formally defined. However it is not possible and practical to define the whole UML notation formally. Compliance levels help to avoid model interchange problems between different

tools by increasing the probability that tools support the same or compatible language subsets. (OMG 2006b.)

The specification of UML AD does not define mapping to any executable language, but the syntax should make mapping possible (OMG 2006b). By extending or customizing the UML Model Driven Architecture (MDA), UML can be mapped to BPEL4WS (Mantell 2005). In that mapping UML AD would be used only for describing the behavior of the BPEL4WS process. Another similar solution is to use the UML Model Driven Development (MDD) approach. In that approach UML AD is first transformed to BPMN and then to BPEL4WS (Kalnins and Vitolins 2006).

4 Workflow Patterns Framework

In this chapter we describe the theoretical background for the workflow patterns framework.

The Workflow Patterns framework (Workflow Patterns 2006) provides a reference framework that can be used to evaluate different workflow modelling notations. It consists of a number of workflow patterns that are common in general process aware information systems. Thus, these patterns can be held as the basic requirements for a business process modelling technique. The basic idea of pattern based analysis is to evaluate how the analysed modelling notation is able to present a given workflow pattern. (van der Aalst et al. 2002; Wohed et al. 2005). This on the other hand means that the work flow patterns are able to say something about the representation power of a notation: it shows if something can be modelled with a notation and if something cannot. Workflow patterns are introduced in the appendix 1.

The patterns given by Workflow Patterns are further divided into three categories: **control-flow patterns**, **data patterns** and **resource patterns**. (van der Aalst et al. 2002; Wohed et al. 2005.)

4.1 Control-flow patterns

Control-flow patterns are patterns that describe the flow of control in systems. Control flow patterns are divided in six categories that are: basic control-flow patterns, advanced branching and synchronization patterns, structural patterns, multiple instance patterns, state based patterns and cancellation patterns. (van der Aalst et al. 2002.)

4.2 Data Patterns

The data patterns are patterns related to data and data-objects. Data patterns are further divided into four categories that are: data visibility patterns, data interaction patterns, data transfer patterns and data-based routing patterns. (Russell et al. 2005a.)

4.3 Resource Patterns

The resource patterns are patterns related to matters concerning the distribution of work to the resources associated with a business process, and the management of this work by those resources. Resource patterns are further divided into seven categories that are: creation patterns, push patterns, pull patterns, detour patterns, auto-start patterns, visibility patterns and multiple resource patterns. (Russell et al. 2005b.)

5 Bunge-Weber-Wand model

In this chapter we provide the theoretical background for the used BWW-model.

The Bunge-Weber-Wand model (later BWW-model) is a meta-model for evaluating modelling techniques that are used to model real life information systems. The BWW-model is based on ontology defined by Mario Bunge (1977) and is developed by Yair Wand and Ron Weber. The main idea behind Bunge's ontology and later the BWW-model is the idea, that there are certain requirements for a modelling notation that has to be fulfilled, in order that the modelling notation is able to represent and capture the needed aspects of the real world system. It is suggested that the BWW-model can be used to analyze the ontological strengths and weaknesses of modelling techniques and notations and further to find out the ontological completeness and the ontological clarity[4] of the evaluated notation at hand. (Rosemann and Green 2002; Recker et al. 2005.) Thus, with the BWW-model one can evaluate the representation power of a notation from one perspective.

The BWW-model's key constructs can be divided in four groups, that are things, states of things, events and transformations and systems, that are build around things (Recker et al. 2005):

1. **Things**, the properties of things and the types of things. Things are the basic unit of the BWW-model. A thing refers to an entity in a real world. Two or more things can in composition be also accounted for a thing. (Rosemann and Green 2002.)

2. **States of things.** State refers to all of the properties of a thing in a given point in time (Rosemann and Green 2002; Recker and Mendling 2006).

3. **Events and transformations that can happen on things.** Events refer to an occurrence that changes the state of a thing. A transformation on the other hand is the mapping between two states of a thing. (Rosemann and Green 2002; Recker and Mendling 2006.)

4. **Systems, that are built around things.** This refers to that things can be composed to systems, which may have subsystems and interfaces to the system's environment (Rosemann and Green 2002; Recker and Mendling 2006).

The BWW-model has been used in over twenty research projects to evaluate the various modelling techniques. For example BPEL, BPMN, ebXML and ECP have been previously evaluated using the BWW-model. (Recker et al. 2005.) There are also studies where the BWW-model has been used in the analysis and evaluation of IS-design

[4] *Ontological completeness* refers to the situation where there is a grammatical notation construct for each ontological construct. *Ontological clarity* is the extent to which the grammar of the notation does not exhibit construct overload, construct redundancy or construct excess. (Green and Rosemann 2000.)

methods, data flow-diagrams, ER-diagrams, NIAM, and the open modelling language (OML) (Opdahl and Henderson-Sellers 2002). However, when evaluating a modelling technique with the BWW-model, it is worth noting, that the BWW-model base evaluation relies heavily on the experiences and the expert-level of the evaluators. (Rosemann and Green 2002.) Thus, the findings gathered using the BWW-model might be somewhat subjective.

6 Workflow patterns framework analysis

A lot of research has been done in the field of workflow patterns. Workflow patterns have been identified and described accurately. (van der Aalst 2002; Russell et al. 2005.) BPMN and UML AD have been analyzed against these patterns. (Wohed et al. 2006a; Wohed et al. 2006b).

Workflow patterns are described in the appendix 1. BPMN and UML AD support for a pattern is marked with '+','-' and '+/-' sings. A '+' indicates direct support for a pattern. A '–' indicates lack of support. A '+/-' indicates that pattern is partially supported. (Wohed et al. 2006a; Wohed et al. 2006b.)

There is no clear answer to the question which notation is better from the workflow pattern's viewpoint. We are trying to highlight the most important differences between BPMN and UML AD that have been found when workflow patterns are analyzed. This will be done by comparing the support for Control-flow, Data and Resource patterns in both notations.

6.1 Control–flow Patterns

Version 2.0 of UML has brought significant enhancements to process modeling with UML AD. The direct support for the control flow patterns is now much better. There are six out of twenty control-flow patterns that were not supported by former versions of UML AD and which are now supported. (Wohed et al. 2004.)

There are different opinions on how control-flow patterns are supported by BPMN and UML AD. White has compared BPMN and UML AD and he found only one control-flow pattern that can not be directly modeled with UML AD but can be modeled with BPMN (2004b). However Wohed et al. have heavily criticized that study and even found mistakes that White has made (2004; 2006b). If White who has been developing BPMN can not model complex business process structures correctly with these notations how can not technically oriented business people?

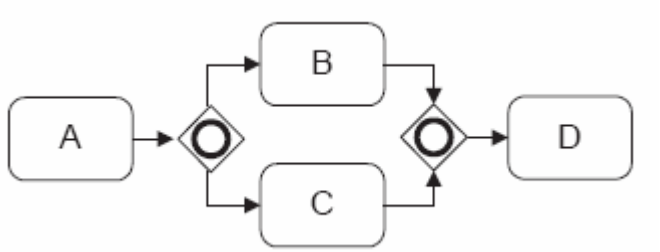
There are not many differences on how BPMN and UML AD support control-flow patterns. Control-flow pattern groups and the support issues are now discussed.

Basic Control–flow

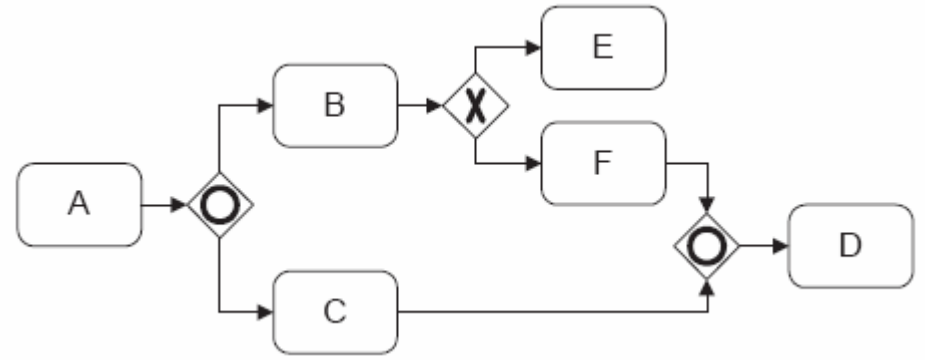
The Basic Control-Flow patterns should be supported by all process modeling languages because they correspond to control-flow constructs that are defined by the Workflow Management Coalition. BPMN and UML AD both support all these patterns. (Wohed et al. 2006b.)

Advanced Synchronization

Synchronizing Merge is not supported by UML AD. It is partially supported by BPMN because only structured workflow is supported. [See figures 13 and 14] (Wohed et al. 2006b.)



Synchronizing Merge in BPMN: structured workflow can be modeled. (Wohed et al. 2006b)
Figure 13



Synchronizing Merge in BPMN: unstructured workflow can not be modeled. This does not work. If activity E is selected Inclusive Gateway (after F) will not know it. (Wohed et al. 2006b)
Figure 14

Structural Patterns, Multiple Instances Patterns and Cancellation Patterns

Structural Patterns, Multiple Instances Patterns and Cancellation Patterns are equally well supported by both notations. (Wohed et al. 2006b.)

State-Based Patterns

The Interleaved Parallel Routing pattern is partially supported only by BPMN. BPMN uses the concept of an Ad-Hoc Process. However Wohed et al. claims that the pattern can be modeled only if the activities are simple tasks. If a sequence of activities should be executed in an arbitrary order the semantics of the Ad-Hoc process are not clear enough. (Wohed et al. 2006b.)

Control-flow patterns did not reveal significant differences between notations. BPMN scored better because it supports partially two patterns that UML AD does not support at

all. However most complex structures are so difficult to model that it is easy to make mistakes. It is easy to draw diagrams that do not work as they should. The specifications of notations should be read carefully to be able to model processes correctly.

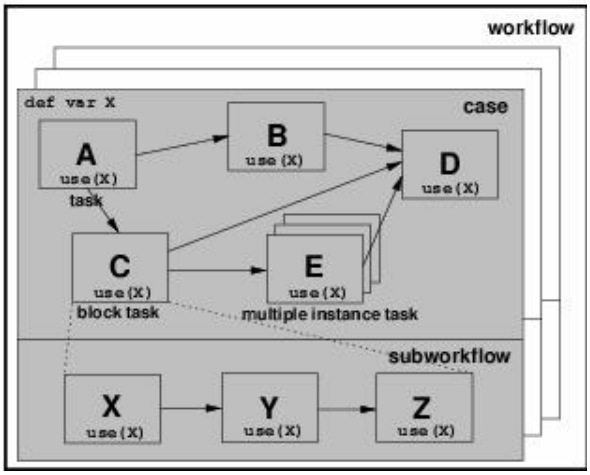
6.2 Data patterns

There are many differences in how BPMN and UML AD support data patterns. However when these 40 data patterns are categorized most of the pattern groups are equally well supported by both notations.

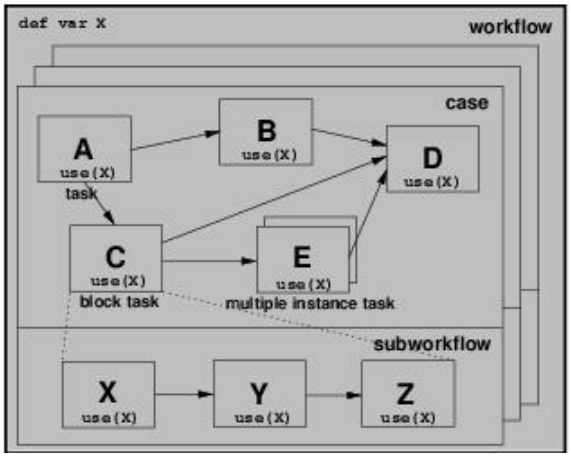
Data Visibility

Data Visibility patterns describe the contexts in which data elements can be defined and utilized (Russell et al. 2005a). We think that a notation does not have to support all of these patterns to be able to effectively define business processes. It is enough to support a comprehensive subset of these patterns. For example there are three patterns that are not supported by BPMN or UML AD. The absence of support for some patterns is more a notation design issue than a serious shortcoming.

Data Visibility is equally well supported by both notations. BPMN supports Case Data and UML AD Workflow Data. BPMN supports only partially Multiple Instance Data and UML AD only partially Task Data (Wohed et al. 2006b). Thus both notations score equally well.



Case level data visibility. Variable X is visible for all tasks throughout a workflow case. (BABEL)
Figure 15



Workflow data visibility. Variable X is visible globally throughout all workflow cases. (BABEL)
Figure 16

Data Interaction

Data elements should be able to pass from component to component in a workflow and also outside the workflow (Russell et al. 2005a). In a workflow UML AD supports data elements passing between a single component and a Multiple Instance Task which BPMN does not support. However BPMN supports much better Data Interaction patterns because UML AD does not support external data interactions at all. (Wohed et al. 2004.) That means UML AD can not pass data elements outside the model. In contrast BPMN can pass data elements from task to environment and vice versa in both push and pull oriented ways. (Wohed et al. 2006b.)

This is a meaningful difference when choosing a modeling notation for business process modeling. If other UML diagrams are also used the relationships between UML AD and other diagrams can be specified. This might help to overcome the gap. Hence if only UML AD is used it means that business processes that should pass data outside the model can not be modeled. For example if data should be stored to a database which is in an external process and not a part of the workflow.

Data Transfer

Data Transfer patterns describe the possible mechanisms that can be used to pass data elements between one workflow component and another (Russell et al. 2005a). UML AD supports data transfer by reference and BPMN supports data transfer both by value and by reference. Additionally UML AD supports directly data transformations when data elements are passed. (Wohed et al. 2006b.) Data Transfer patterns do not reveal significant differences between notations.

Data-based Routing

Data-based Routing patterns describe how data can affect the execution of the workflow (Russell et al. 2005a). Both notations support pre- and post conditions for tasks but BPMN does not support conditions that are based on data value. Only conditions that are based on the existence of data are supported by BPMN. Both notations support tasks that are executed when some trigger fires them. UML AD does not support Data Based Task Triggers, which means that tasks can not be triggered by data elements. (Wohed et al. 2006b.)

BPMN directly supports more Data Patterns than UML AD. There are eight patterns that are directly supported by BPMN and not by UML AD but only five patterns vice versa. That is not a big difference but BPMN clearly supports Data patterns better than UML AD. The slight difference comes from the support for data interactions. UML AD supports workflow level data interactions better but it does not support external data interactions at all.

6.3 Resource Patterns

BPMN and UML AD support the same resource patterns (Wohed et al. 2006b). No differences can be drawn from resource patterns between these notations. Actually there are only a few resource patterns that are supported. According to the BPMN specification modeling resources will not be a part of BPMN (OMG 2006a). UML specification does not give a clear answer to why resource patterns are not supported in UML AD, but there are other diagram types that are probably better suitable for modeling resources (OMG 2006b).

There are some resource patterns that are supported. However that is more like a consequence than planned design issue. For example both notations use Pools and Lanes constructs. Each pool indicates the resource that will be responsible for executing a set of activities. This is how direct and role based resource allocation are directly supported in both notations. However that kind of restrictive manner in which Swimlanes are specified denies the support for many other resource patterns. Only the name of the Swimlane can be specified but the relationships between distinct Swimlanes can not. That is why for example capacity-based resource allocation can not be modeled. (Wohed et al. 2006b.)

6.4 Workflow patterns conclusion

The Workflow patterns framework revealed some differences in the representation power between the BPMN and UML AD notations. BPMN succeeded better with control-flow and data patterns. Both notations give equally weak support for resource patterns.

When considering representation power we think that the only difference that really matters is the fact that UML AD does not support the external data interactions. However other differences might also matter for example when complex processes should be modeled. Then a small difference in support for control-flow patterns might be a significant one.

7 Bunge-Weber-Wand model analysis

Now we are going to look at BPMN and UML AD through the lens of the BWW-model that was described earlier. Because of the fact that the BWW-model analysis requires a lot of modelling experience with the notations that are analyzed, we restrict our study only to the findings given by earlier studies on the matter. We try to figure out, if these findings are relevant in the business process modelling paradigm and further, try to generate practical examples from them.

7.1 BWW-model and BPMN

Many of the previous analyses and research done by using the BWW-model have had something to do with business process modelling, such as the often cited article by Green and Rosemann (2000) in which the authors analyze the five views of ARIS with BWW-model. While the analyses with BWW-model are numerous, it seems that there are only a few analyses that analyse solely BPMN. Therefore our discussion here is based on the work done by Recker et al. (2006).

According to the findings of Recker et al. (2006) the BWW-model analyses reveal nine cases, where BPMN lack either ontological clarity or ontological completeness.[5] Some of these findings clearly show, that all the real world concepts and phenomena cannot be captured using the BPMN. Recker et al. grouped these findings into four categories that are Construct deficit, Construct redundancy, Construct excess and Construct overload.

Construct deficits suggest, that the notation of BPMN is not able to model all the concepts of the real world. These include the following problems:

- One is not able to represent state, stable state, unstable state, conceivable state space, state law, lawful state space or conceivable event space with BPMN models. This means, that business rules requiring states and related state transformations will have some unclearness, when they are modelled using BPMN.
- One is not able to represent history with BPMN models. This means, that the modelling of business processes that relate to the recovery and reliability of interacting process entities will be problematic using BPMN.
- One is not able to represent system structure with BPMN. This can lead to problems when one is modelling with inter-organizational business processes that require system structure.

(Recker et al. 2006.)

Construct redundancy means that for one real-world concept, there might be many representations in the used notation. These problems are:

[5] Ontological completeness refers to the situation where there is a grammatical notation construct for each ontological construct. Ontological clarity is the extent to which the grammar of the notation does not exhibit construct overload, construct redundancy or construct excess. (Green and Rosemann 2000.)

- A thing can be represented in BPMN either by a Pool or by a Lane. This might be confusing for the end-user who might not be sure, which to use. Is an organizational department, for example, a Pool or a Lane?
- A transformation can be represented in BPMN by Activity, Task, Collapsed Sub-Process, Expanded Sub-Process, Nested Sub-Process or Transaction. This might be confusing for the end-user who might not be sure, which to use.
- An event can be represented by nine different BPMN events. This might be confusing to the end-user, who might not be sure which event does what.

(Recker et al. 2006.)

According to Recker et al. (2006) construct excess means that there are some constructs in BPMN that bear no meaning in the real-world model of the BWW-model. Link, Off-Page-Connector, Association Flow and Activity Looping, taking just a few for example, appear to be unnecessary from the BWW point of view.

Construct overload means that some constructs in the notation can represent many real-world BWW-concepts. These problems are:

- In BPMN the Lane construct can represent, for example, thing, class, kind, system or subsystem in the BWW-model. This means that the end-user of the model has to know something extra about the modelled system in order to completely understand the BPMN model at hand.
- As the BPMN construct Lane, also Pool maps to thing, class, kind, system, subsystem etc. causing the same problems that were discussed with the Lane construct.

(Recker et al. 2006.)

7.2 BWW-model and UML AD

When considering the BWW-model with UML AD, there is one thing first to be found out: there has not beet much research considering this subject. The general UML class diagrams have had some analysis against BWW-model (Evermann and Wand 2001; Opdahl and Henderson-Sellers 2002), but only a few studies truly handle UML AD from the ontological perspective. Our discussion here is based on the work done by Dussart et al. (2002).

According to the findings of Dussart et al. (2002) there can also be found cases in the UML AD notation, where the representation power of the notation is not sufficient enough to fulfil all the requirements given by the BWW-model.

The first problem that UML AD faces, are the states in the BWW-model. Although, UML AD is able to represent a state as a state of the object in an activity diagram, the notation does not support stable state, unstable state, conceivable state space, state law, lawful state space or conceivable event space. (Dussart et al. 2002.) Thus, nearly the same problem arises here as with BPMN.

As BPMN, also the UML AD is not able to model history related to a business process. Also, system structures defined by the BWW-model are not supported by UML AD. (Dussart et al. 2002.)

Dussart et al. (Dussart et al. 2002) further suggest that UML AD faces construct overload, where some constructs, in this case Activity, can represent several phenomena in the BWW-model. An Activity can, for example, represent a transformation, a process, an event or a property, making it harder for the end-user to distinguish what construct to use. Dussart et al. also found, that a Swimlane in UML AD can represent either a thing or a property of a thing.

In UML AD there can be found also construct excess, for there are no real-world BWW-constructs for the branching construct in UML AD (Dussart et al. 2002).

7.3 How do BMPN and UML AD Differ When Evaluated With BWW-model?

On the whole, the BWW-model analyses against these two notations result in quite the similar results. Both notations have problems in modelling states, system structures and history, as seen in two previous chapters. Though, it is worth noting, that according to Dussart et al. (2002) UML AD supports the basic construct of state, which is totally omitted in the BPMN by the terms of Recker et al. (2006).[6]

Also, construct excess and construct overload are present in both notations. The construct overload even refers to the same problem in BPMN and UML AD: Pool (BPMN), Lane (BPMN) and Swimlane (UML AD) can all represent many things in the BWW-model construct. Thing, class, kind, system, subsystem can all be represented either by Pool, Lane or Swimlane.

The most crucial difference found between these notations seems to be construct redundancy, which was not found in UML AD. These findings might suggest, that the UML AD notation could be more comprehensible to the end-user, because it has only one construct for one real-life object.

In the end, the ontological differences and differences in representation power of the notations at hand seem to be quite narrow: from the ontological point of view BPMN and UML AD can be found similar in many respects. There are some differences that would prefer the use of UML AD, but there is not enough research on this matter that proper conclusions could be made. While some would find UML AD better in representation power because of the lack of construct redundancy, others might find that these kinds of details do not have any relevance.

[6] When considering states one should be careful. The state model given in the BWW-model could be interpreted to have many things in common with the traditional state-machine-modelling.

In the Table 2 below we summarize the ontological attributes and differences of these two notations. The letter x in the table means, that the notation has a problem with this area.

Table 2
BPMN and UML AD ontological differences: ontological problems

Problem	BPMN	UML AD
Cannot represent state	x	
Cannot represent stable state, unstable state, conceivable state space, state law, lawful state space or conceivable event space	x	x
Cannot represent history	x	x
Cannot represent system structure	x	x
Has some construct excess	x	x
Has some construct redundancy	x	
Has some construct overload	x	x

8 Concluding Remarks

Results from the Workflow patterns framework analysis in chapter six showed that BPMN has a better representation power in control flow and data -patterns. While both notations give equally weak results in resource patterns, the only real and meaningful difference seems to be the lack of external interactions support in UML AD. In this light the representation power of BPMN can be seen better than UML AD.

Results from the BWW-model analysis in chapter seven were not very considerable. In many respects the BWW-model analyses against these two notations have very similar results. Depending on the viewpoint, UML AD could be seen better in representation power, but only little.

When recapturing this research, the differences between the representation power of BPMN and UML AD are very narrow. On the whole, BPMN could be seen to have only a little more representation power than UML AD, because of the better Workflow pattern analysis results.

Thus, we could conclude, that if there should be a need to choose which notation to use, the decision can not be done based on the representation power. It is better to base the choice between the notations on other matters like existing experience on some tool and notation or equal. This is not, unfortunately, in the scope of this research.

9 Further Discussion

As stated in the concluding remarks, the representation power may not be the best approach to analyze and compare business process modeling notations. During our study we have noticed that the biggest problems might not concern the notations, but the used tools instead. It was, for example, especially hard to find a tool that supports all the elements that are specified in the specifications. Also, the support for some Workflow patterns and the representation power of the notation does not help if you can not model the pattern with any tool.

The above suggests that in further studies that compare notations, some other viewpoint might be more effective. This could be also concluded from the interview research done on the real modellers and end-users of business process modelling notations by Recker et al. (2005). According to Recker et al. in many cases end users found the ontological (and thus representation power) differences between notations irrelevant.

In our view, optional viewpoints for further research could be usability and ease of learning of the notation, tool support or hands-on experience.

Although the purpose of this study was to examine differences between these two notations, it is worth noting for that our hands-on-feeling of the maturity of the notations is good. On the whole, both of the notations seem to be quite mature in general terms.

GLOSSARY

Activity	An activity is a generic term for work that a company or an organization performs via business processes.
Artifact	An Artifact is a graphical object that provides supporting information about the Process or elements within the Process.
BPD	A Business Process Diagram is a part of the Business Process Modeling Notation. It represents visually the process with elements specified in the notation.
BPEL4WS	The Business Process Execution Language for Web Services is an orchestration language that can directly be used while creating web services. The Business Process Modeling Notation is possible transform to this orchestration language.
BPMN	The Business Process Modeling Notation is a notation for modeling business processes. It is has private, abstract and collaboration levels of abstraction.
Business Process	A business process is a description of tasks and outcomes associated with a business activity. The business process is often drawn, depicting tasks, roles, resources and actions to be taken according to the business needs.
BWW-model	The Bunge-Weber-Wand-model is a meta-model for evaluating modelling techniques that are used to model real life information systems.
Notation	A Notation consists of the visual appearance of the graphical elements and the semantics of the elements.
Representation Power	Representation power is a notation ability to represent real world phenomena.
Process Model	A process model is a description of a process at the type level.
Specification	A Specification is an actual specification document of a given notation.
Specification Scope	Specification scope is a planned scope of what for the notation is suitable, according to the specification.

UML AD	UML 2.0 Activity Diagrams are typically used for business process modeling, for modeling the logic captured by a single use case or usage scenario, or for modeling the detailed logic of a business rule.
Workflow Pattern	A Workflow Pattern is a generalization of an issue that happens in a business process. Where the Workflow Patterns framework provides a reference framework that can be used to evaluate different workflow modelling notations.

REFERENCES

Babel. 15 November 2006.
http://www.bpm.fit.qut.edu.au/projects/babel/

Bunge, Mario A. 1977. Treatise on Basic Philosophy Volume3: Ontology I – The Furniture of the World. Kluwer Academic Publisher, Dordrecht.

Dussart, Aymeric; Aubert, Benoit A.; Patry, Michel 2002. An Evaluation of Inter-Organizational Workflow Modeling Formalisms. Cahier du GReSI no 02-06 Août 2002.

Evermann, Joerg; **Wand**, Yair 2001. Towards Ontologically Based Semantics for UML Constructs A Chapter In the Book: Conceptual Modeling - ER 2001: 20th International Conference on Conceptual Modeling, Yokohama, Japan, November 27-30, 2001. Proceedings

Green, Peter; **Rosemann**, Michael 2000. Integrated Process Modeling: An Ontological Evaluation. Information Systems, Volume 25, Number 2, April 2000, pp. 73-87(15)

Kalnins, Audris; **Vitolins**, Valdis 2006. Use of UML and Model Transformations for Workflow Process Definitions.

Mantell, Keith 22 September 2005. From UML to BPEL. http://www-http://128.ibm.com/developerworks/webservices/library/ws-uml2bpel/

Object Management Group. 2006a. Business Process Modeling Notation Specification. OMG Final Adopted Specification dtc/06-02-01. February 2006

Object Management Group. 2006b. Unified Modeling Language: Superstructure. version 2.0. formal/05-07-04. August 2005

Opdahl, Andreas L.; **Henderson-Sellers**, Brian 2002. Ontological Evaluation of the UML Using the Bunge–Wand–Weber Model. Software and Systems Modeling, Volume 1, Number 1 / September, 200.2

Recker, Jan; Indulska, Marta; Rosemann, Michael; Green, Peter 2005. Do Process Modelling Tecniques Get Better? A Comparative Ontological Analysis of BPMN. 16th Australian Conference on Information Systems. 29 Nov – 2 Dec 2005, Sydney.

Recker, Jan; Indulska, Marta; Rosemann, Michael; Green, Peter 2006. How Good Is BPMN Really? Insights From Theory and Practise. Proceedings 14th European Conference on Information Systems. Goeteborg, Sweden.

Recker, Jan; **Mendling**, Jan 2006. On the Translation between BPML and BPEL: Conceptual Mismatch between Process Modeling Languages.

Rosemann, Michael; **Green,** Peter 2002. Developing a meta model for the Bunge–Wand–Weber ontological constructs. Information Systems, Vol. 27, Issue 2, April 2002, 75–91.

Russell, Nick; ter Hofstede, Arthur H.M; Edmond, David 2005a. Workflow Data Patterns. Proc. of 24th Int. Conf. on Conceptual Modeling (ER05).

Russell, Nick; ter Hofstede, Arthur H.M; Edmond, David 2005b. Workflow Resource Patterns. Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE '05).

Russell, Nick; van der Aalst, Wil M.P; ter Hofstede, Arthur H.M; Wohed, Petia 2006. On the Suitability of UML 2.0 Activity Diagrams for Business Process Modeling.

Schnieders, Arnd., Puhlmann, Frank. and Weske, Mathias. 2004. Process Modeling Techniques. PESOAReport No. 01/2004. Hasso Plattner Institute.

van der Aalst, Wil M.P; ter Hofstede, Arthur H.M; Kiepuszewski, B; Barros A.B. 2002. Workflow Patterns. BPM Center Report BPM.

Webster, Jane; **Watson**, Richard T. 2002. Analyzing the Past to Prepare for the Future: Writing a Literature Review. MIS Quartely Vol. 26, Issue 2, June 2002, xiii-xxiii.

White, Stephen. 2004a. Introduction to BPMN. IBM Corporation.

White, Stephen. 2004b. Process Modeling Notations and Workflow Patterns. In L. Fischer, editor, Workflow Handbook 2004, pages 265–294. Future Strategies Inc., Lighthouse Point, FL, USA.

Wikipedia. Activity diagram, 30 October 2006. http://en.wikipedia.org/wiki/Activity_diagram

Wohed, Petia; van der Aalst, Wil M.P; Dumas, Marlon; ter Hofstede, Arthur H.M; Russell, Nick. (2004) Pattern-based Analysis of UML Activity Diagrams, BETA Working Paper Series, WP 129, Eindhoven University of Technology, Eindhoven.

Wohed, Petia; van der Aalst, Wil M.P; Dumas, Marlon; ter Hofstede, Arthur H.M; Russel, Nick 2005. Pattern-based Analysis of UML Activity Diagrams.

Wohed, Petia; van der Aalst, Wil M.P; Dumas, Marlon; ter Hofstede, Arthur H.M; Russell, Nick. 2006a. On the Suitability of BPMN for Business Process Modelling. In Dustdar, Schahram and Fiadeiro, Jose-Luis and Sheth, Amit, Eds. Proceedings 4th International Conference on Business Process Management, pages pp. 161-176, Vienna, Austria.

Wohed, Petia; van der Aalst, Wil M.P; Dumas, Marlon; ter Hofstede, Arthur H.M; Russell, Nick. 2006b. Pattern-based Analysis of BPMN – An extensive evaluation of the Control-flow, the Data and the Resource Perspectives (revised version). BPM Center Report BPM-06-17, BPMcenter.org.

Workflow Patterns. 15 November 2006.
http://www.workflowpatterns.com/

APPENDIX 1

Control-flow patterns and their descriptions are copied from (http://www.workflowpatterns.com)
Data and Resource patterns and their descriptions are copied from BABEL (http://www.bpm.fit.qut.edu.au/projects/babel/)

In table 3 BPMN and UML AD support for a pattern is marked with '+','-' and '+/-' sings. A '+' indicates direct support for a pattern. A '–' indicates lack of support. A '+/-' indicates that pattern is partially supported. (Wohed et al. 2006a; Wohed et al. 2006b.)

Table 3
Control-flow patterns (Workflow Patterns 2006)

Control-flow patterns	Description	BPMN	UML AD
Basic Control-flow Patterns			
Sequence	An activity in a workflow process is enabled after the completion of another activity in the same process.	+	+
Parallel split	A point in the workflow process where a single thread of control splits into multiple threads of control which can be executed in parallel, thus allowing activities to be executed simultaneously or in any order.	+	+
Synchronization	A point in the workflow process where multiple parallel subprocesses/activities converge into one single thread of control, thus synchronizing multiple threads. It is an assumption of this pattern that each incoming branch of a synchronizer is executed only once.	+	+
Exclusive choice	A point in the workflow process where, based on a decision or workflow control data, one of several branches is chosen.	+	+
Simple merge	A point in the workflow process where two or more alternative branches come together without synchronization. It is an assumption of this pattern that none of the alternative branches is ever executed in parallel.	+	+
Advanced Branching and Synchronization patterns			
Multiple Choice	A point in the workflow process where, based on a decision or workflow control	+	+

	data, a number of branches are chosen.		
Synchronizing merge	A point in the workflow process where multiple paths converge into one single thread. If more than one path is taken, synchronization of the active threads needs to take place. If only one path is taken, the alternative branches should reconverge without synchronization. It is an assumption of this pattern that a branch that has already been activated, cannot be activated again while the merge is still waiting for other branches to complete.	+/-	-
Multiple Merge	A point in a workflow process where two or more branches reconverge without synchronization. If more than one branch gets activated, possibly concurrently, the activity following the merge is started *for every activation of every incoming branch*.	+	+
Discrimination	The discriminator is a point in a workflow process that waits for one of the incoming branches to complete before activating the subsequent activity. From that moment on it waits for all remaining branches to complete and "ignores" them. Once all incoming branches have been triggered, it resets itself so that it can be triggered again (which is important otherwise it could not really be used in the context of a loop).	+	+
Structural patterns			
Arbitrary cycles	A point in a workflow process where one or more activities can be done repeatedly.	+	+
Implicit termination	A given subprocess should be terminated when there is nothing else to be done. In other words, there are no active activities in the workflow and no other activity can be made active (and at the same time the workflow is not in deadlock).	+	+
Multiple Instances Patterns			
Multiple Instances without Synchronization	Within the context of a single case (i.e., workflow instance) multiple instances of an activity can be created, i.e., there is a facility to spawn off new threads of control. Each of these threads of control is independent of other threads. Moreover, there is no need to synchronize these	+	+

	threads.		
Multiple with a Priori Design Time Knowledge	For one process instance an activity is enabled multiple times. The number of instances of a given activity for a given process instance is known at design time. Once all instances are completed some other activity needs to be started.	+	+
Multiple with a Priori Runtime Knowledge	For one case an activity is enabled multiple times. The number of instances of a given activity for a given case varies and may depend on characteristics of the case or availability of resources, but is known at some stage during runtime, before the instances of that activity have to be created. Once all instances are completed some other activity needs to be started.	+	+
Multiple without a Priori Runtime Knowledge	For one case an activity is enabled multiple times. The number of instances of a given activity for a given case is not known during design time, nor is it known at any stage during runtime, before the instances of that activity have to be created. Once all instances are completed some other activity needs to be started.	-	-
State-based Patterns			
Deferred Choice	A point in the workflow process where one of several branches is chosen. In contrast to the XOR-split, the choice is not made explicitly (e.g. based on data or a decision) but several alternatives are offered to the environment. However, in contrast to the AND-split, only one of the alternatives is executed. This means that once the environment activates one of the branches the other alternative branches are withdrawn. It is important to note that the choice is delayed until the processing in one of the alternative branches is actually started, i.e. the moment of choice is as late as possible.	+	+
Interleaved Parallel Routing	A set of activities is executed in an arbitrary order: Each activity in the set is executed, the order is decided at run-time, and no two activities are executed at the same moment (i.e. no two activities are active for the same workflow instance at the same time).	+/-	-

Milestone	The enabling of an activity depends on the case being in a specified state, i.e. the activity is only enabled if a certain milestone has been reached which did not expire yet. Consider three activities named A, B, and C. Activity A is only enabled if activity B has been executed and C has not been executed yet, i.e. A is not enabled before the execution of B and A is not enabled after the execution of C.	-	-
Cancellation Patterns			
Cancel activity	An enabled activity is disabled, i.e. a thread waiting for the execution of an activity is removed.	+	+
Cancel case	A case, i.e. workflow instance, is removed completely (i.e., even if parts of the process are instantiated multiple times, all descendants are removed).	+	+

Table 4
Data patterns (Babel 2006)

Data patterns	Description	BPMN	UML AD
Data Visibility Patterns			
Task data	Data elements can be defined by tasks which are accessible only within the context of individual execution instances of that task.	+	+/-
Block data	Block tasks (i.e. tasks which can be described in terms of a corresponding sub-workflow) are able to define data elements which are accessible by each of the components of the corresponding sub-workflow.	+	+
Scope data	Data elements can be defined which are accessible by a subset of the tasks in a case.	-	-
Multiple instance data	Tasks which are able to execute multiple times within a single workflow case can define data elements which are specific to an individual execution instance.	+/-	+
Case data	Data elements are supported which are specific to a process instance or case of a workflow. They can be accessed by all components of the workflow during the	+	-

	execution of the case.		
Folder data	Data elements can be defined which are accessible by multiple cases on a selective basis.	-	-
Workflow data	Data elements are supported which are accessible to all components in each and every case of the workflow and are within the control of the workflow system.	-	+
Environment data	Data elements which exist in the external operating environment are able to be accessed by components of the workflow during execution.	-	-
Data interaction patterns (Internal)			
Task to Task	The ability to communicate data elements between one task instance and another within the same case.	+	+
Block Task to Sub-Workflow Decomposition	The ability to pass data elements from a block task instance to the corresponding sub-workflow that defines its implementation.	+	+
Sub-Workflow Decomposition to Block Task	The ability to pass data elements from the underlying sub-workflow back to the corresponding block task instance.	+	+
to Multiple Instance Task	The ability to pass data elements from a preceding task instance to a subsequent task which is able to support multiple execution instances. This may involve passing the data elements to all instances of the multiple instance task or distributing them on a selective basis.	-	+
from Multiple Instance Task	The ability to pass data elements from a task which supports multiple execution instances to a subsequent task.	-	+
Case to Case	The passing of data elements from one case of a workflow during its execution to another case that is executing concurrently.	-	-
Data interaction patterns (External)			
Task to Environment (Push-Oriented)	The ability of a task to initiate the passing of data elements to a resource or service in the operating environment.	+	-
Environment to Task (Pull-Oriented)	The ability of a workflow task to request data elements from resources or services in the operational environment.	+	-
Environment to Task	The ability for a workflow task to receive	+	-

(Push-Oriented)	and utilise data elements passed to it from services and resources in the operating environment on an unscheduled basis.		
Task to Environment (Pull-Oriented)	The ability of a workflow task to receive and respond to requests for data elements from services and resources in the operational environment.	+	-
Case to Environment (Push-Oriented)	The ability of a workflow case to initiate the passing of data elements to a resource or service in the operational environment.	-	-
Environment to Case (Pull-Oriented)	The ability of a workflow case to request data from services or resources in the operational environment.	-	-
Environment to Case (Push-Oriented)	The ability of a workflow case to accept data elements passed to it from services or resources in the operating environment.	-	-
Case to Environment (Pull-Oriented)	The ability of a workflow case to respond to requests for data elements from a service or resource in the operating environment.	-	-
Workflow to Environment (Push-Oriented)	The ability of a workflow engine to pass data elements to resources or services in the operational environment.	-	-
Environment to Workflow (Pull-Oriented)	The ability of a workflow to request workflow-level data elements from external applications.	-	-
Environment to Workflow (Push-Oriented)	The ability of services or resources in the operating environment to pass workflow-level data to a workflow process.	-	-
Workflow to Environment (Pull-Oriented)	The ability of a workflow engine to handle requests for workflow-level data from external applications.	-	-
Data transfer patterns			
Data transfer by value – incoming	The ability of a workflow component to receive incoming data elements by value relieving it from the need to have shared names or common address space with the component(s) from which it receives them.	+	-
Data transfer by value – outgoing	The ability of a workflow component to pass data elements to subsequent components as values relieving it from the need to have shared names or common address space with the component(s) to which it is passing them.	+	-
Data transfer - copy in/out	The ability of a workflow component to copy the values of a set of data elements into its address space at the commencement	+/-	-

	of execution and to copy their final values back at completion.		
Data transfer by reference – without lock	The ability to communicate data elements between workflow components by utilising a reference to the location of the data element in some mutually accessible location. No concurrency restrictions apply to the shared data element.	-	-
Data transfer by reference – with lock	The ability to communicate data elements between workflow components by passing a reference to the location of the data element in some mutually accessible location. Concurrency restrictions are implied with the receiving component receiving the privilege of read-only or dedicated access to the data element.	+	+
Data transformation – input	The ability to apply a transformation function to a data element prior to it being passed to a workflow component.	+/-	+
Data transformation – output	The ability to apply a transformation function to a data element immediately prior to it being passed out of a workflow component.	+/-	+
Data-based routing patterns			
Task precondition – data existence	Data-based preconditions can be specified for tasks based on the presence of data elements at the time of execution.	+	+
Task precondition - data value	Data-based preconditions can be specified for tasks based on the value of specific parameters at the time of execution.	-	+
Task postcondition – data existence	Data-based postconditions can be specified for tasks based on the existence of specific parameters at the time of execution.	+	+
Task postcondition – data existence	Data-based postconditions can be specified for tasks based on the value of specific parameters at the time of execution.	-	+
Event-based task trigger	The ability for an external event to initiate a task.	+	+
Data-based task trigger	The ability to trigger a specific task when an expression based on workflow data elements evaluates to true.	+	-
Data-based routing	The ability to alter the control flow within a workflow case as a consequence of the value of data-based expressions.	+	+

Table 5
Resource patterns (Babel 2006)

Resource patterns	Description	BPMN	UML AD
Creation patterns			
Direct Allocation	The ability to specify at design time the identity of the resource that will execute a task.	+	+
Role-Based Allocation	The ability to specify at design time that a task can only be executed by resources which correspond to a given role.	+	+
Deferred Allocation	The ability to defer specifying the identity of the resource that will execute a task until runtime	-	-
Authorization	The ability to specify the range of resources that are authorised to execute a task.	-	-
Separation of Duties	The ability to specify that two tasks must be allocated to different resources in a given workflow case.	-	-
Case Handling	The ability to allocate the work items within a given workflow case to the same resource.	-	-
Retain Familiar	Where several resources are available to undertake a work item, the ability to allocate a work item within a given workflow case to the same resource that undertook a preceding work item.	-	-
Capability-based Allocation	The ability to offer or allocate instances of a task to resources based on specific capabilities that they possess.	-	-
History-based Allocation	The ability to offer or allocate work items to resources on the basis of their previous execution history.	-	-
Organizational Allocation	The ability to offer or allocate instances of a task to resources based their position within the organisation and their relationship with other resources.	-	-
Automatic Execution	The ability for an instance of a task to execute without needing to utilise the services of a resource.	+	+
Push patterns			
Distribution by Offer-Single Resource	The ability to offer a work item to a selected individual resource.	-	-
Distribution by Offer-Multiple Resources	The ability to offer a work item to a group of selected resources.	-	-
Distribution by	The ability to directly allocate a work item	+	+

Allocation-Single Resource	to a specific resource for execution.		
Random Allocation	The ability to offer or allocate work items to suitable resources on a random basis.	-	-
Round Robin Allocation	The ability to allocate a work item to available resources on a cyclic basis.	-	-
Shortest Queue	The ability to allocate a work item to the resource that has the least number of work items allocated to it.	-	-
Early Distribution	The ability to advertise and potentially allocate work items to resources ahead of the moment at which the work item is actually enabled for execution.	-	-
Distribution on Enablement	The ability to advertise and allocate work items to resources at the moment they are enabled for execution.	+	+
Late Distribution	The ability to advertise and allocate work items to resources after the work item has been enabled.	-	-
Pull patterns			
Resource-Initiated Allocation	The ability for a resource to commit to undertake a work item without needing to commence working on it immediately.	-	-
Resource-Initiated Execution (Allocated Work Item)	The ability for a resource to commence work on a work item that is allocated to it.	-	-
Resource-Initiated Execution (Offered Work Item)	The ability for a resource to select a work item offered to it and commence work on it immediately.	-	-
System-Determined Work Queue Content	The ability of the workflow engine to order the content and sequence in which work items are presented to a resource for execution.	-	-
Resource-Determined Work Queue Content	The ability for resources to specify the format and content of work items listed in the work queue for execution.	-	-
Selection Autonomy	The ability for resources to select a work item for execution based on its characteristics and their own preferences.	-	-
Detour patterns			
Delegation	The ability for a resource to allocate a work item previously allocated to it to another resource.	-	-
Escalation	The ability of the workflow system to offer or allocate a work item to a resource or	-	-

	group of resources other than those it has previously been offered or allocated to in an attempt to expedite the completion of the work item.		
Deallocation	The ability of a resource (or group of resources) to relinquish a work item which is allocated to it and make it available for allocation to another resource or group of resources.	-	-
Stateful Reallocation	The ability of a resource to allocate a work item to another resource without loss of state data.	-	-
Stateless Reallocation	The ability for a resource to reallocate a work item currently being executed to another resource without retention of state.	-	-
Suspension/Resumption	The ability for a resource to suspend and resume execution of a work item.	-	-
Skip	The ability for a resource to skip a work item allocated to it and mark the work item as complete.	-	-
Redo	The ability for a resource to redo a work item that has previously been completed in a case.	-	-
Pre-Do	The ability for a resource to execute a work item ahead of the time that it has been offered or allocated to resources working on a given case.	-	-
Auto-start patterns			
Commencement on Creation	The ability for a resource to commence execution on a work item as soon as it is created.	+	+
Commencement on Allocation	The ability to commence execution on a work item as soon as it is allocated to a resource.	-	-
Piled Execution	The ability of the workflow system to initiate the next instance of a workflow task (perhaps in a different case) once the previous one has completed.	-	-
Chained Execution	The ability of the workflow engine to automatically start the next work item in a case once the previous one has completed.	+	+
Visibility Patterns			
Configurable Unallocated Work Item Visibility	The ability to configure the visibility of unallocated work items by workflow participants.	-	-
Configurable Allocated	The ability to configure the visibility of	-	-

Work Item Visibility	allocated work items by workflow participants.		
Multiple Resource Patterns			
Simultaneous Execution	The ability for a resource to execute more than one work item simultaneously.	+	+
Additional Resources	The ability for a given resource to request additional resources to assist in the execution of a work item that they are currently undertaking.	-	-