On the Suitability of BPMN for Business Process Modelling*

P. Wohed[1]**, W.M.P. van der Aalst[2,3], M. Dumas[3], A.H.M. ter Hofstede[3], N. Russell[3]

[1] The Department of Computer and Systems Sciences, SU/KTH, Sweden
petia@dsv.su.se
[2] Faculty of Information Technology, QUT, Australia
{m.dumas, a.terhofstede, n.russell}@qut.edu.au
[3] Department of Technology Management, TUE, The Netherlands
w.m.p.v.d.aalst@tm.tue.nl

Abstract. In this paper we examine the suitability of BPMN for business process modelling, using the Workflow Patterns as an evaluation framework. The Workflow Patterns are a collection of patterns developed for assessing control-flow, data and resource capabilities in the area of Process Aware Information Systems (PAIS). In doing so, we provide a comprehensive evaluation of the capabilities of BPMN, and its strengths and weaknesses when utilised for business process modelling. The analysis provided for BPMN is part of a larger effort aiming at an unbiased and vendor-independent survey of the suitability and the expressive power of some mainstream process modelling languages. It is a sequel to an analysis series where languages like BPEL and UML 2.0 A.D are evaluated.

1 Introduction

The focus on Process-Aware Information Systems (PAIS) during the last decade has led up to a new generation of languages and tools for process description. Existing (mainstream) languages for IS development have recently been revised, e.g. UML 2.0 Activity Diagrams (AD), and new languages like BPMN and BPEL4WS have been developed and have rapidly spread. The common feature of these three languages is their focus on providing a powerful and standardized notation for (business) process modelling. Among the differences it can be mentioned that while UML AD and BPMN are graphical but not-formalised notations, BPEL4WS is an executable language (and therefore, in principle, also formalized) which lacks a graphical notation.

These rough characteristics do not, however, provide any insights into i) the expressive capacity of the languages, ii) their suitability for (business) process modeling, or iii) how they actually relate to each other. To address these issues, a thorough analysis of each of the languages is necessary. In this paper we focus on BPMN. Through a detailed examination, we aim to expose the advantages and shortcomings of BPMN, challenging in this way its suitability for business process modeling and providing comparative insights into it. The analysis of BPMN is a part of a survey on mainstream process modeling languages. It is a sequel to a series of analyses which include analyses of UML 2.0

* This work is funded in part by Interop NoE IST-508011 and by ARC DP0451092.
** Research conducted during a visit at the Queensland University of Technology.

AD [8] and BPEL4WS [1]. The goal of the survey is to provide comparative insights, which is achieved by analysing the languages through one and the same framework.

As a reference analysis framework we use the *Workflow Patterns Framework*[4], a taxonomy of generic, recurring constructs originally devised to evaluate workflow systems, but also suitable to evaluate workflow standards, business process languages and process-aware information systems in general. In accordance with Jablonski and Bussler's original classification [3], these patterns span the *Control-flow*, *Data* and *Resource* perspectives of PAIS. Our choice of this evaluation framework is based on the fact that it is 1) widely used, 2) well accepted, 3) comprehensible to the IT practitioner, 4) at a sufficiently detailed level of abstraction to provide a comprehensive basis for assessing the capabilities of business process modelling languages and 5) the most complete and powerful framework for this form of assessment currently in existence.

In essence, the contributions of this paper are as follows:

- It is the first multi-perspective evaluation of the expressive capabilities of BPMN;
- It provides an assessment of the overall suitability of BPMN for business process modelling;
- It identifies several areas of potential future improvements of BPMN to further strengthen its use for this purpose;
- It provides results that enable comparative analysis of BPMN with competitive languages.

Previous evaluations [10, 6] have analysed the quality and ontological standard of BPMN. The evaluation in [10] is based on the *Semiotic Quality Framework* and positioned by its authors as a fairly general and the Semiotic Quality Framework as complementary to the *Bunge Wand and Weber (BWW) Framework* used in [6]. Relaying on an ontology for Information Systems, the BWW Framework is further at a higher level of generality compared to the Workflow Patterns Framework which was specifically tailored for the assessment of Process Aware Information Systems. Lastly, there has also been a review of the capabilities of the Control-flow perspective of BPMN [13] with the Workflow Patterns Framework. However the limited focus of [13], as well as ambiguities in the results reported in [13] justify the work presented here.

In the remainder of the paper we evaluate BPMN from the three perspectives, starting by the Control-flow, and followed by the Data and Resource perspectives. Then we discuss our findings and compare these with evaluations of UML 2.0 AD and BPEL.

2 The Control-Flow Perspective in BPMN

In this section we examine the control-flow perspective of BPMN and its ability to represent a series of twenty common control-flow modelling requirements that occur when defining process models. These requirements are described in terms of the Workflow Control-flow Patterns [2]. The material in this section summarises the findings reported in [15]. There has also been a review of this perspective of BPMN by White [13], who is one of BPMN's developers. The results reported here differ from these in [13], however due to space limitation for a detailed discussion on the differences we refer to [15].

[4] See `www.workflowpatterns.com` for comprehensive details.

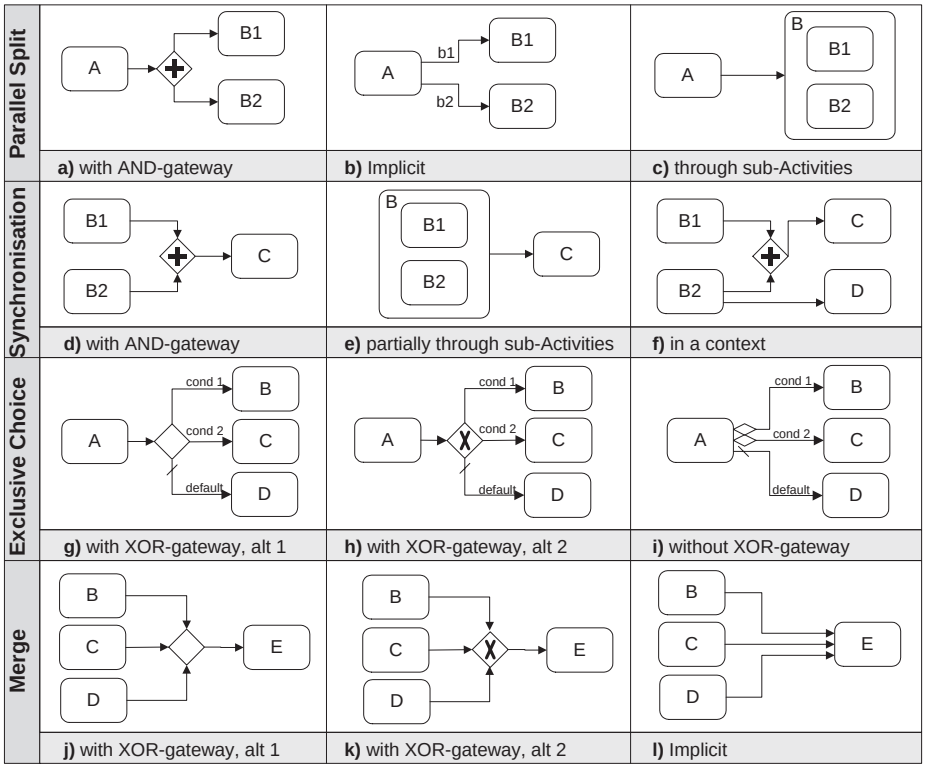

Fig. 1. Basic control-flow patterns in BPMN

2.1 Basic control-flow patterns

The basic control-flow patterns define elementary aspects of process control. These are analogous to the definitions of elementary control-flow concepts laid down by the Workflow Management Coalition [11]. There are five of these patterns:

- CP1: *Sequence* – the ability to depict a sequence of activities;
- CP2: *Parallel split* – the ability to capture a split in a single thread of control into multiple threads of control which can execute in parallel;
- CP3: *Synchronisation* – the ability to capture a convergence of multiple parallel subprocesses/activities into a single thread of control thus synchronising multiple threads;
- CP4: *Exclusive choice* – the ability to represent a decision point in a workflow process where one of several branches is chosen;
- CP5: *Simple merge* – the ability to depict a point in the workflow process where two or more alternative branches come together without synchronisation.

All five of these patterns can be captured in BPMN. Sequence is represented through a sequence flow between two activities. The representation of the remaining of these patterns is illustrated in Figure 1.

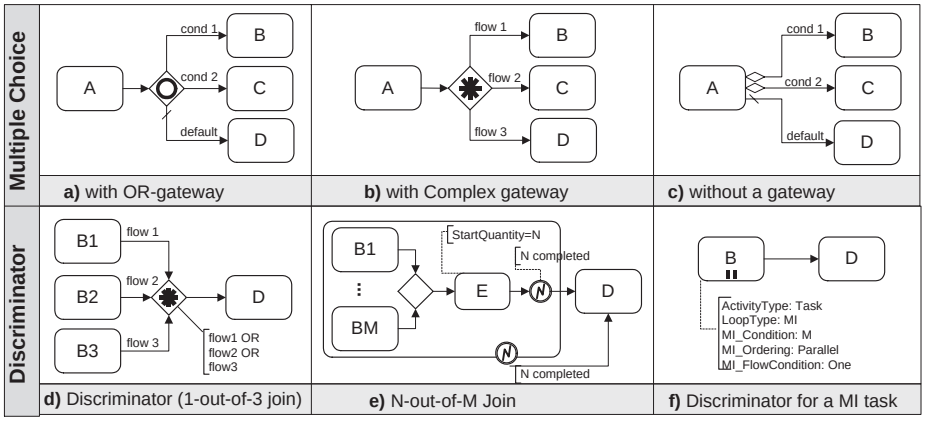

Fig. 2. Advanced branching and synchronisation patterns in BPMN

2.2 Advanced branching and synchronisation patterns

This class of patterns corresponds to advanced branching and synchronisation scenarios that often do not have direct realisations in PAIS but are relatively common in real-life business processes. There are four of these patterns:

- CP6: *Multiple choice* – the ability to represent a divergence of the thread of control into several parallel branches on a selective basis;
- CP7: *Synchronising merge* – the ability to depict the synchronised convergence of two or more alternative branches;
- CP8: *Multiple merge* – the ability to represent the unsynchronised convergence of two or more distinct branches. If more than one branch is active, the activity following the merge is started for every activation of every incoming branch;
- CP9: *Discriminator* – the ability to depict the convergence of two or more branches such that the first activation of an incoming branch results in the subsequent activity being triggered and subsequent activations of remaining incoming branches are ignored. It is a special case of *N-out-of-M* pattern, where N is equal to one.

The multiple choice, multiple merge and discriminator patterns can be captured directly in BPMN. The solution for the multiple merge pattern is identical to the solution for the simple merge pattern (see figures 1j, 1k and 1l). The solutions for the multiple choice and the discriminator patterns are illustrated in Figure 2. The synchronising merge pattern is captured partially through the OR-join gateway. The solution is partial because it assumes a structured workflow context.

2.3 Structural patterns

Structural patterns identify whether the modelling formalism has any restrictions in regard to the way in which processes can be structured (particularly in terms of the type of loops supported and whether a single terminating node is necessary). There are two of these patterns:

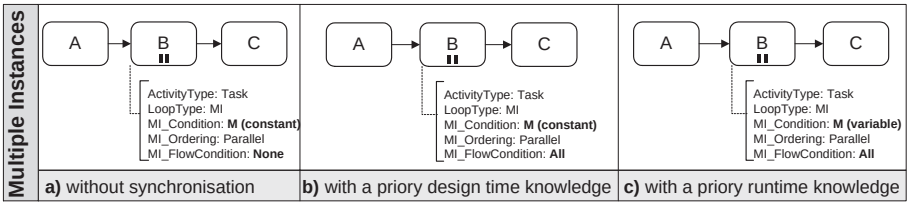

Fig. 3. Multiple Instances patterns in BPMN

- CP10: *Arbitrary cycles* – the ability to represent loops in a process that have multiple entry or exit points;
- CP11: *Implicit termination* – the ability to depict the notion that a given subprocess should be terminated when there are no remaining activities to be completed (i.e. no explicit unique termination node is needed).

Both of these patterns are directly supported in BPMN.

2.4 Multiple Instances (MI) patterns

This class of patterns relates to situations where there can be more than one instance of an activity active at the same time for the same process instance. There are four of these patterns:

- CP12: *MI without synchronisation* – the ability to initiate multiple instances of an activity within a given process instance;
- CP13: *MI with a priori design time knowledge* – the ability to initiate multiple instances of an activity within a given process instance. The number of instances is known at design time. Once all instances have completed, a subsequent activity is initiated;
- CP14: *MI with a priori runtime knowledge* – the ability to initiate multiple instances of an activity within a given process instance. The number of instances varies but is known at runtime before the instances must be created. Once all instances have completed, a subsequent activity is initiated;
- CP15: *MI without a priori runtime knowledge* – the ability to initiate multiple instances of an activity within a given process instance. The number of instances varies but is not known at design time or at runtime before the instances must be created. Once all instances have completed, a subsequent activity is initiated. New instances can be created even while other instances are executing or have already completed.

The first three of these patterns can be captured in BPMN as illustrated in Figure 3. The MI without a priori runtime knowledge pattern is not directly supported in BPMN.

2.5 State-based patterns

This class of patterns characterise scenarios in a process where subsequent execution is determined by the state of the process instance. There are three such patterns:

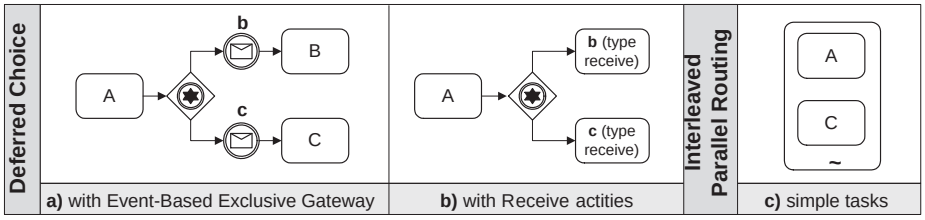

Fig. 4. State-based patterns in BPMN

– CP16: *Deferred choice* – the ability to depict a divergence point in a process where one of several possible branches should be activated. The actual decision on which branch is activated is made by the environment and is deferred to the latest possible moment;
– CP17: *Interleaved parallel routing* – the ability to depict a set of activities that can be executed in arbitrary order;
– CP18: *Milestone* – the ability to depict that a specified activity cannot be commenced until some nominated state is reached which has not expired yet.

Owing to the absence of the notion of state, only the deferred choice pattern can be fully captured in BPMN. This is illustrated in figures 4a and 4b. The interleaved parallel routing pattern is captured for the case when the activities to be interleaved are simple tasks. This solution is illustrated in Figure 4c. The milestone pattern is not supported.

2.6 Cancellation patterns

Cancellation patterns characterise the ability of the modelling formalism to represent the potential termination of activities and process instances in certain (specified) circumstances. There are two such patterns:

– CP19: *Cancel activity* – the ability to depict that an enabled activity should be disabled in some nominated circumstance;
– CP20: *Cancel case* – the ability to represent the cancellation of an entire process instance (i.e. all activities relating to the process instance) in some nominated circumstance.

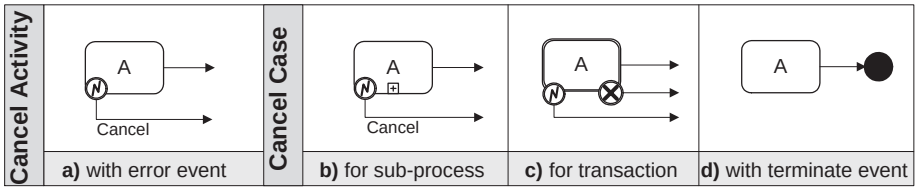

Fig. 5. Cancellation patterns in BPMN

	1	2	3	4		1	2	3	4
Basic Control–flow					11. Implicit Termination	+	+	+	+
1. Sequence	+	+	+	+	Multiple Instances Patterns				
2. Parallel Split	+	+	+	+	12. MI without Synchronization	+	+	+	+
3. Synchronisation	+	+	+	+	13. MI with a priori Design Time Knowledge	+	+	+	+
4. Exclusive Choice	+	+	+	+	14. MI with a priori Runtime Knowledge	+	+	–	+
5. Simple Merge	+	+	+	+	15. MI without a priori Runtime Knowledge	–	–	–	+/–
Advanced Synchronisation					State-Based Patterns				
6. Multiple Choice	+	+	+	+	16. Deferred Choice	+	+	+	+
7. Synchronising Merge	+/–	–	+	+	17. Interleaved Parallel Routing	+/–	–	+/–	–
8. Multiple Merge	+	+	–	–	18. Milestone	–	–	–	+/–
9. Discriminator	+	+	–	–	Cancellation Patterns				
Structural Patterns					19. Cancel Activity	+	+	+	+/–
10. Arbitrary Cycles	+	+	–	–	20. Cancel Case	+	+	+	+

Table 1. Support for the Control–flow Patterns
in **1**–BPMN, **2**–UML2.0 AD [16, 8], **3**–BPEL4WS [14, 1], and **4**–Oracle BPEL PM v.10.1.2 [4]

Both of these patterns can be captured in BPMN. The solutions are shown in Figure 5.

Table 1 summarises the results from this part of the evaluation. The table also shows the results from the evaluations of UML 2.0 AD, BPEL4WS, and a concrete system based on the latter language Oracle BPEL Process Manager (PM) Version 10.1.2. In the conclusion we will compare BPMN with these other languages.

3 The Data Perspective in BPMN

Extensions [9] to the *Workflow Patterns Initiative* have focused on identifying and defining generic constructs that occur in the the data perspective of PAIS. In total forty data patterns have been delineated in four distinct groups – data visibility, data interaction, data transfer and data-based routing. In this section, an analysis of BPMN is presented using the data patterns described in [9].

3.1 Data visibility patterns

Data visibility patterns seek to characterise the various ways in which data elements can be defined and utilised within the context of a process. In general, this is determined by the main construct to which the data element is bound as it implies a particular scope in which the data element is visible and capable of being utilised. There are eight patterns which relate to data visibility:

- DP1: *Task data* – data elements defined and accessible in the context of individual execution instances of a task or activity;
- DP2: *Block data* – data elements defined by block tasks (i.e. tasks which can be described in terms of a corresponding decomposition) and accessible to the block task and all corresponding components within the associated decomposition;
- DP3: *Scope data* – data elements bound to a subset of the tasks in a process instance;
- DP4: *Multiple instance data* – data elements specific to a single execution instance of a task (where the task is able to be executed multiple times);

– DP5: *Case data* – data elements specific to a process instance which are accessible to all components of the process instance during execution;
– DP6: *Folder data* – data elements bound to a subset of the tasks in a process definition but accessible to all task instances regardless of the case to which they correspond;
– DP7: *Workflow data* – data elements accessible to all components in all cases;
– DP8: *Environment data* – data elements defined in the operational environment which can be accessed by process elements.

BPMN supports several of these patterns. Task, Block and Case data are supported through the attribute Properties of Task, Sub-Process and Process elements, respectively. Scope data is not supported as the Group construct does not offer any data handling for the elements it groups together.

Multiple instance data is partially supported. There are three situations where multiple instances of a given task may arise: (1) Where a task is specifically designated as having multiple instances in the process model. The lack of any data attributes (akin the attribute Properties) for the distinct instances of a multiple instances activity eliminates the possibility to handle any instance specific data for a multiple instances task. (2) Where a task can be triggered multiple times, e.g., it is part of a loop or it is a task following after a multiple merge construct. These situations are allowable in BPMN. (3) Where two tasks share the same decomposition. This is supported. An activity decomposition can be captured through the notion of an Independent Sub-Process. Several Independent Sub-Processes can invoke one and the same Process.

Folder, Workflow and Environment data patterns are not supported in BPMN.

3.2 Data interaction patterns

Data interaction patterns deal with the various ways in which data elements can be passed between components within a process instance and also with the operating environment (e.g. data transfer between a component of a process and an application, data store or interface that is external to the process). They examine how the characteristics of the individual components can influence the manner in which the trafficking of data elements occurs. There are six internal data interaction patterns:

– DP9: *Data elements flowing between task instances*;
– DP10: *Data elements flowing to a block*;
– DP11: *Data elements flowing from a block*;
– DP12: *Data elements flowing to a multiple instance task instance*;
– DP13: *Data elements flowing from a multiple instance task instance*;
– DP14: *Data elements flowing between process instances or cases*.

Data interaction between tasks (DP9) can be utilized in three different ways, namely: (i) through integrated control and data channels or; (ii) through distinct control and data channels or; (iii) through the support of global shared data. BPMN supports global shared data (through the Properties attribute for a Process), hence the third alternative is clearly supported. It also appears that the first two alternatives are supported. Data interaction through distinct control and data channels is supported through the notion of

Data Object. Data interaction through integrated control and data channels is supported through the construct of Data Objects associated with sequence flows.

Furthermore, the data interactions block task to and from sub-workflow (DP10 and DP11) are directly supported. One of the means of doing this is via parameters, i.e., through the Input- and OutputPropertyMaps attributes. This is relevant for the cases when the decomposition is defined through an Independent Sup-Process. Another way for doing this is through the global shared data defined for a process. This is relevant for the cases when the decomposition is defined through an Embedded Sup-Processes.

Data interaction to and from multiple instance task instances (DP12 and DP13) and data interaction between cases (DP14) are not supported in BPMN.

In addition to the internal data interaction patterns, there are 12 external data interaction patterns. These are characterised by three dimensions:

- The type of process element – task, case or complete process – that is interacting with the environment;
- Whether the interaction is push or pull-based;
- Whether the interaction is initiated by the process component or the environment.

The patterns *Task to Environment, Push* and *Pull*, and *Environment to Task, Push* and *Pull*, (i.e., DPs 15, 16, 17 and 18) are supported in BPMN. They are captured through one or a pair of message flow(s) flowing to, from, or to and from a task and the boundary of a pool representing the environment. Note that for these patterns the environment is modelled explicitly.

The patterns Case to Environment, Push and Pull, as well as Environment to Case, Push and Pull (i.e., DPs 19–22) are not supported. Message flows can indeed be drawn between the boundaries of two pools where one of the pools represents a process and the other one the environment. However, "If the Message Flow is connected to the boundary to the Expanded Sub-Process, then this is equivalent to connecting to the Start Event for incoming Message Flow or the End Event for outgoing Message Flow." ([12],p. 117). Hence, this construct does not provide support for data exchange of case data at *any* moment during the execution of a case.

Finally, the *Workflow to Environment, Push* and *Pull*, and *Environment to Workflow, Push* and *Pull* patterns (i.e., DPs 23-26) are not supported, as workflow data is not supported in BPMN (see DP7 above).

3.3 Data transfer patterns

Data transfer patterns focus on the way in which data elements are actually transferred between one process element and another. They aim to capture the various mechanisms by which data elements can be passed across the interface of a process element. There are seven distinct patterns in this category:

- DP27: *Data transfer by value– incoming* – incoming data elements passed by value;
- DP28: *Data transfer by value– outgoing* – outgoing data elements passed by value;
- DP29: *Data transfer– copy in/copy out* – where a process element synchronises data elements with an external data source at commencement and completion;

– DP30: *Data transfer by reference– without lock* – data elements are communicated between components via a reference to a data element in some mutually accessible location. No concurrency restrictions are implied;
– DP31: *Data transfer by reference– with lock* – similar to DP30 except that concurrency restrictions are implied with the receiving component receiving the privilege of read-only or dedicated access to the data element;
– DP32: *Data transformation– input* – where a transformation function is applied to a data element prior to it being passed to a subsequent component;
– DP33: *Data transformation– output* – where a transformation function is applied to a data element prior to it being passed from a previous component.

In BPMN, the DP27 and DP28 patterns are supported through the notion of the Input and OutputSets. DP29 Data transfer – copy in/copy out is partially supported. It occurs when a decomposition is realised with Independent Sub-Processes. The data attributes to be copied into/out of the Independent Sub-Process are specified through the Input- and OutputPropertyMaps attributes. As these PropertyMaps are in the form of Expressions we assume that also different transformation functions can be captured through them. This implies that patterns DP32 and DP33 are also partially supported. The support is considered to be partial because it only applies to data transfer to and from an Independent Sub-Process, and not between any couple of Activities.

Finally, the DP31 data transfer by reference – with lock is supported. As BPMN adopts a token-oriented approach to data passing, the parameters – which typically relate to objects – are effectively consumed at activity commencement and only become visible and accessible to other activities once the specific activity to which they were passed has completed and returned them.

3.4 Data-based routing patterns

Data-based routing patterns capture the various ways in which data elements can interact with other perspectives and influence the overall execution of the process. There are seven (relatively self-explanatory) patterns in this category:

– DP34: *Task precondition – data existence*;
– DP35: *Task precondition – data value*;
– DP36: *Task postcondition – data existence*;
– DP37: *Task postcondition – data value*;
– DP38: *Event-based task trigger*;
– DP39: *Data-based task trigger*;
– DP40: *Data-based routing*.

BPMN does not directly support pre- and postcondition definitions. Hence, the patterns 35 and 37 are not supported. In the cases data transfer is realised though Data Objects, the boolean attributes RequiredForStart and ProducedAtCompletion capture the pre- and postconditions for data existence (i.e., DP34 and DP36).

The Message, Timer, Error and Cancel Event constructs provide direct support for the event-based task triggering pattern (DP38). The Rule Event construct provides support for the Data-based task trigger pattern (DP39). Finally, the Data-based routing

Data Visibility	1	2	3	4	Data Interaction (External) (cont.)	1	2	3	4
1. Task Data	+	+/–	+/–	+/–	21. Env. to Case – Push-Oriented	–	–	–	–
2. Block Data	+	+	–	–	22. Case to Env. – Pull-Oriented	–	–	–	–
3. Scope Data	–	–	+	+	23. Workflow to Env. – Push-Oriented	–	–	–	–
4. Multiple Instance Data	+/–	+	–	+/–	24. Env. to Workflow – Pull-Oriented	–	–	–	–
5. Case Data	+	–	+	+	25. Env. to Workflow – Push-Oriented	–	–	–	–
6. Folder Data	–	–	–	–	26. Workflow to Env. – Pull-Oriented	–	–	–	–
7. Workflow Data	–	+	–	–	Data Transfer				
8. Environment Data	–	–	+	+	27. by Value – Incoming	+	–	+	+
Data Interaction (Internal)					28. by Value – Outgoing	+	–	+	+
9. between Tasks	+	+	+	+	29. Copy In/Copy Out	+/–	–	–	+
10. Block Task to Sub-wf Decomp.	+	+	–	–	30. by Reference – Unlocked	–	–	+	+
11. Sub-wf Decomp. to Block Task	+	+	–	–	31. by Reference – Locked	+	+	+/–	–
12. to Multiple Instance Task	–	+	–	+/–	32. Data Transformation – Input	+/–	+	–	–
13. from Multiple Instance Task	–	+	–	+/–	33. Data Transformation – Output	+/–	+	–	–
14. Case to Case	–	–	+/–	–	Data-based Routing				
Data Interaction (External)					34. Task Precondition – Data Exist.	+	+	+/–	–
15. Task to Env. – Push-Oriented	+	–	+	+	35. Task Precondition – Data Val.	–	+	+	+
16. Env. to Task – Pull-Oriented	+	–	+	+	36. Task Postcondition – Data Exist.	+	+	–	–
17. Env. to Task – Push-Oriented	+	–	+/–	+	37. Task Postcondition – Data Val.	–	+	–	–
18. Task to Env. – Pull-Oriented	+	–	+/–	+	38. Event-based Task Trigger	+	+	+	+
19. Case to Env. – Push-Oriented	–	–	–	–	39. Data-based Task Trigger	+	–	+/–	–
20. Env. to Case – Pull-Oriented	–	–	–	–	40. Data-based Routing	+	+	+	+

Table 2. Support for the Data Patterns

in **1**–BPMN, **2**–UML2.0 AD [16, 8], **3**–BPEL4WS [14, 1], and **4**–Oracle BPEL PM v.10.1.2 [4]

(DP40) is supported, as Condition Expressions are possible to specify for Sequence Flows.

Table 2 shows a summary of the results from the Data perspective.

4 The Resource Perspective in BPMN

Recent work [7] has focused on the resource perspective and the manner in which work is distributed amongst and managed by the resources associated with a business process. Our investigations have indicated that these patterns are relevant to all forms of PAIS including modelling languages such as XPDL and business process enactment languages such as BPEL4WS. In this section, we examine the resource perspective of BPMN and its expressive power in regard to work distribution.

Forty three workflow resource patterns have been identified in seven distinct groups:

- *Creation patterns* – which correspond to restrictions on the manner in which specific work items can be advertised, allocated and executed by resources;
- *Push patterns* – which describe situations where a PAIS proactively offers or allocates work to resources;

– *Pull patterns* – which characterise scenarios where resources initiate the identification of work that they are able to undertake and commit to its execution;
– *Detour patterns* – which describe deviations from the normal sequence of state transitions associated with a business process either at the instigation of a resource or the PAIS;
– *Auto-start patterns* – which relate to situations where the execution of work is triggered by specific events or state transitions in the business process;
– *Visibility patterns* – which describe the ability of resources to view the status of work within the PAIS;
– *Multiple resource patterns* – which describe scenarios where there is a many-to-many relationship between specific work items and the resources undertaking those work items.

In BPMN, the association of a particular action or set of actions with a specific resource is illustrated through the use of the Pools and Lanes constructs, commonly called Swimlanes ([12],pp. 102-106). "A Pool represents a Participant in the Process. A Participant can be a specific business entity (e.g. a company) or can be a more general business role." ([12],p. 103). "A Lane is a sub–partition within a Pool..." ([12],p. 106). Hence, the direct allocation pattern (RP1) as well as the role–based allocation pattern (RP2) are directly supported. Furthermore, a partitioning of a Process into Pools and Lanes is not required, i.e., the resource allocation for the different activities is not necessarily done during design time. This means that automatic execution pattern (RP11) is also supported in BPMN.

None of the other Creation Patterns are supported within BPMN. This is a consequence of the restrictive manner in which Swimlanes are specified (i.e., only by specifying their Names, and in case of sub-division, the sub-division hierarchy) and the lack of support for relationships between distinct Swimlanes. Lack of a capability specification, integrated authorisation framework, organisational model and access to some execution history, rules out any form of support for capability–based allocation (RP8), the authorisation (RP4), organisational allocation (RP9) and history-based allocation (RP5, RP6 and RP7) patterns respectively.

The execution semantics of a BPMN model is based on Petri Net token flow, hence activities become "live" once they receive the specified StartQuantity control-flow tokens. The resource associated with a given Swimlane can have multiple activities executing at the same time. There is no notion of scheduling work execution or of resources selecting the work (i.e. the activity) they wish to undertake, hence there is minimal support for the Push, Auto-start or Multiple Resource patterns. The following patterns from these classes are directly supported:

– RP14: *Distribution by allocation - single resource* – the resource(s) associated with a given Swimlane is immediately allocated a Task/Sub-Process once it is triggered.
– RP19: *Distribution on enablement* – all activities in a Swimlane are associated with the resource responsible for the Swimlane when they are triggered.
– RP36: *Commencement on creation* – an activity is assumed to be live as soon as it receives the specified StartQuantity control-flow tokens.

Creation Patterns	1	2	4	Pull Patterns (cont.)	1	2	4
1. Direct Allocation	+	+	+	24. System-Determ. Work Queue Content	–	–	–
2. Role-Based Allocation	+	+	+	25. Resource-Determ. Work Queue Content	–	–	+
3. Deferred Allocation	–	–	+	26. Selection Autonomy	–	–	+
4. Authorization	–	–	–	Detour Patterns			
5. Separation of Duties	–	–	–	27. Delegation	–	–	+
6. Case Handling	–	–	+	28. Escalation	–	–	+
7. Retain Familiar	–	–	+	29. Deallocation	–	–	+
8. Capability-based Allocation	–	–	+	30. Stateful Reallocation	–	–	+
9. History-based Allocation	–	–	+/–	31. Stateless Reallocation	–	–	–
10. Organizational Allocation	–	–	+/–	32. Suspension/Resumption	–	–	+
11. Automatic Execution	+	+	+	33. Skip	–	–	+
Push Patterns				34. Redo	–	–	–
12. Distribution by Offer-Single Resource	–	–	+	35. Pre-Do	–	–	–
13. Distribution by Offer-Multiple Resources	–	–	+	Auto-start Patterns			
14. Distribution by Allocation-Single Resource	+	+	+	36. Commencement on Creation	+	+	–
15. Random Allocation	–	–	+/–	37. Commencement on Allocation	–	–	–
16. Round Robin Allocation	–	–	+/–	38. Piled Execution	–	–	–
17. Shortest Queue	–	–	+/–	39. Chained Execution	+	+	–
18. Early Distribution	–	–	–	Visibility Patterns			
19. Distribution on Enablement	+	+	+	40. Config. Unallocated Work Item visibility	–	–	–
20. Late Distribution	–	–	–	41. Config. Allocated Work Item visibility	–	–	–
Pull Patterns				Multiple Resource Patterns			
21. Resource-Init. Allocation	–	–	–	42. Simultaneous Execution	+	+	+
22. Resource-Init. Exec. - Allocated Work Item	–	–	+	43. Additional Resources	–	–	+
23.Resource-Init. Execution - Offered Work Item	–	–	+				

Table 3. Support for the Resource Patterns

in **1**–BPMN, **2**–UML2.0 AD [16, 8], and **4**–Oracle BPEL PM v.10.1.2 [4][5]

- RP39: *Chained execution* – once an activity is completed, subsequent activity(ies) receive a control-flow token and is(are) triggered immediately when the specified StartQuantity of tokens is reached.
- RP42: *Simultaneous execution* – there are no constraints on how many instances of a task specified for one Swimlane can be active at any time.

None of the Pull, Detour or Visibility patterns are supported. The results from this part of the evaluation are summarised in Table 3.

5 Conclusions

There are inherent difficulties in applying the Workflow Patterns Framework for assessing a language that does not have a commonly agreed upon formal semantics nor an execution environment. Indeed, the BPMN specification [12] provides a mapping from BPMN to BPEL4WS. Also, BPMI.org claims that more than 30 vendors have

[5] BPEL does not cover the resource perspective and is therefore omitted from Table 3.

implemented BPMN. However, the mapping to BPEL4WS is only partial (see [5] for discussions on this) and most of the vendors support just a small subset of the language reported in [12] assuming also different semantics when executing it. All ambiguities caused by the lack of formalisation and all assumptions made because of this were therefore carefully documented during our work.

The results from the Workflow Patterns analysis of BPMN are presented in tables 1, 2 and 3. A "+" indicates a direct support for a pattern, a "+/-" indicates a partial support, and a "-" indicates lack of support. For the purposes of comparison, these tables also contain the results from the evaluations of UML 2.0 AD (reprinted from [16] and [8]), as well as the results for BPEL4WS (cf. [14, 1]) and Oracle's implementation of BPEL4WS in Oracle BPEL PM Version 10.1.2 (cf. [4]), obtained during earlier phases of this work. From these tables it can be seen that BPMN provides support for the majority of the control-flow patterns, for nearly half of the data patterns and only very limited support for the resource patterns.

For the control-flow perspective (see Table 1) it can be seen that the Multiple Instances without a Priory Runtime Knowledge and Milestone patterns are not captured. Moreover, the patterns Synchronizing Merge and Interleaved Parallel Routing are only partially captured. The limitations in capturing two of the State-based patterns is determined by the lack of the state concept in BPMN. Furthermore, the semantics of the OR-join Gateway needs to be generalized in order to be able to capture the Synchronizing Merge in a non-structured workflow context. Finally, the MI activity construct needs to be extended in order to capture all of the Multiple Instances patterns.

An outcome from the analysis of the Control-flow perspective which is not visible from Table 1 is that many of the patterns, have several representations. Most of the simplest patterns have as many as tree different representations in BPMN. Furthermore, detailed knowledge of the non-graphically represented attributes of the modeling constructs of BPMN is required for capturing some of the patterns.

For the support of BPMN in the Data perspective (see Table 2) it can be seen that Workflow and Environment Data are not supported. Data interaction to and from a Multiple Instances task is not supported because any instance specific data for the multiple instances of a task can not be defined. Also the support for the external data interaction patterns is very limited. Only the patterns capturing the interaction between tasks and environment are supported, as they can be captured by modeling the environment as an external process.

Finally, the support of BPMN in the Resource perspective is minor (see Table 3). It is indeed stated in the specification ([12],p.22) that the modeling of organizational structures and resources is outside the scope for BPMN. However, the presence of the concepts Lane and Pool for representing participants and roles gives the impression that some issues with resource allocation of activities were (at least partially) addressed in the language. It is obvious, though, that Pools and Lanes do not provide a means for representing the subtleties associated with selective work allocation across a range of possible resources and the management of those work items at run-time and that more work remains for adequately capturing this perspective in BPMN.

Regarding the comparison of BPMN with UML 2.0 AD and BPEL, a few observations can be made. In the Control-flow perspective, BPMN and UML 2.0 AD are almost

fully overlapping. BPMN is slightly stronger when it comes to the representation of the CP17 Interleaved Parallel Routing and the CP7 Synchronising Merge patterns, but these are only minor differences. By comparing BPMN to BPEL4WS in this perspective, it can be observed that some patterns are supported in BPMN but not in BPEL4WS and vise versa. This implies that the appearance of these patterns in a BPMN model would require special care when translating the model into BPEL4WS. A translation from BPMN to BPEL is, hence, not as straightforward as it often is believed to be.

For the Data perspective the support for the patterns in BPMN and UML 2.0 AD is slightly different. UML 2.0 AD is stronger in capturing multiple instances data as well as data interaction to and from multiple instances tasks, while BPMN is stronger in the data interaction between task and environment (due to the fact that the environment can be explicitly modelled). There are further differences for the patterns in the Data Transfer and Data-based Routing categories, as well as discrepancies from the patterns captured by BPEL. However, even if the set of patterns captured in this perspective is distinct for every language and even if none of the languages fully captures all the patterns, it can be argued that the Data perspective is reasonably well covered.

Unfortunately, this can not be stated for the Resource perspective, the support of which is on the whole not present in BPEL4WS and only minor in BPMN and UML 2.0 AD. This is contrasted by the support in Oracle BPEL PM, which, as an execution environment, provides more or less support for almost two thirds of the patterns. The presence of concepts specific to the Resource perspective (e.g., Lanes and Pools in BPMN and Partitions in UML 2.0 AD) reveals the needs (and the intention) of these languages for supporting this perspective. However, providing support for a minimal set of patterns only exposes the immaturity in adequately covering this perspective. The support for the Resource perspective in Oracle BPEL PM confirmers the necessity and the importance of capturing this perspective for executable process definitions. It also exposes the gap between the representation capacity of contemporary process analysis and modeling languages from the current process execution languages. In order to achieve consistent and coherent process models from the analysis phase to their implementation and enactment this gap needs to be filled. Instead of creating yet others process modelling notations overlapping in their control-flow capabilities, we should rather focus on further developing the existing notations to satisfactory cover all the aspects relevant for PAIS.

Acknowledgement: Thanks to Chun Ouyang, for valuable discussions on BPMN during the work.

References

1. W.M.P. van der Aalst, M. Dumas, A. H.M. ter Hofstede, N. Russell, H. M.W Verbeek, and P. Wohed. Life After BPEL? In M. Bravetti et al., editor, *In Proc. of the 2nd Int. Workshop on Web Services and Formal Methods (WS-FM)*, volume 3670 of *LNCS*, pages 35–50. Springer Verlag, 2005.
2. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

3. S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, London, UK, 1996.
4. N.A. Mulyar. Pattern-based Evaluation of Oracle-BPEL (v.10.1.2). Technical report, Center Report BPM-05-24, BPMcenter.org, 2005.
5. C. Ouyang, M. Dumas, S. Breutel, and A.H.M. ter Hofstede. Translating Standard Process Models to BPEL. To appear in *Proceedings of 18th International Conference on Advanced Information Systems Engineering (CAiSE 2006)*, June 2006.
6. J. Recker, M. Indulska, M. Rosemann, and P. Green. Do Process Modelling Techniques Get Better? A Comparative Ontological Analysis of BPMN. In B. Campbell, J. Underwood, and D. Bunker, editors, *16th Australasian Conference on Information Systems*. Australasian Chapter of the Association for Information Systems, Sydney, Australia, 2005.
7. N. Russell, W.M.P. van der Aalst, A.H.M. ter Hofstede, and D. Edmond. Workflow Resource Patterns: Identification, Representation and Tool Support. In O. Pastor and J. Falcão e Cunha, editors, *Proc. of 17th Int. Conf. on Advanced Information Systems Engineering (CAiSE05)*, volume 3520 of *LNCS*, pages 216–232. Springer, 2005.
8. N. Russell, W.M.P. van der Aalst, A.H.M. ter Hofstede, and P. Wohed. On the Suitability of UML 2.0 Activity Diagrams for Business Process Modelling. In *to appear in Proc. of The 3rd Asia-Pacific Conf. on Conceptual Modelling (APCCM 2006)*. Springer Verlag, Jan 2006.
9. N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Workflow Data Patterns. In L. Delcambre et al., editor, *Proc. of 24th Int. Conf. on Conceptual Modeling (ER05)*, volume 3716 of *LNCS*, pages 353–368. Springer Verlag, Oct 2005.
10. T. Wahl and G. Sindre. An Analytical Evaluation of BPMN Using a Semiotic Quality Framework. In Jaelson Castro and Ernest Teniente, editors, *CAiSE'05 Workshops. Volume 1*, pages 533–544. FEUP, Porto, Portugal, 2005.
11. WfMC. Workflow Management Coalition Terminology & Glossary, Document Number WFMC-TC-1011, Document Status - Issue 3.0. Technical report, Workflow Management Coalition, Brussels, Belgium, 1999.
12. S. White. Business Process Modeling Notation (BPMN). Version 1.0 - May 3, 2004, BPMI.org, 2004. `www.bpmi.org`.
13. S. White. Process Modeling Notations and Workflow Patterns. In L. Fischer, editor, *Workflow Handbook 2004*, pages 265–294. Future Strategies Inc., Lighthouse Point, FL, USA, 2004.
14. P. Wohed, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. Analysis of Web Services Composition Languages: The Case of BPEL4WS. In Il-Y. Song et al., editor, *Proc. of 22nd Int. Conf. on Conceptual Modeling (ER 2003)*, volume 2813 of *LNCS*, pages 200–215. Springer, 2003.
15. P. Wohed, W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, and N. Russell. Pattern-based Analysis of BPMN - an extensive evaluation of the Control-flow, the Data and the Resource Perspectives. BPM Center Report BPM-05-26, BPMcenter.org, 2005.
16. P. Wohed, W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, and N. Russell. Pattern-Based Analysis of the Control-Flow Perspective of UML Activity Diagrams. In L. Delcambre et al, editor, *Proc. of 24th Int. Conf. on Conceptual Modeling (ER05)*, volume 3716 of *LNCS*, pages 63–78. Springer Verlag, 2005.