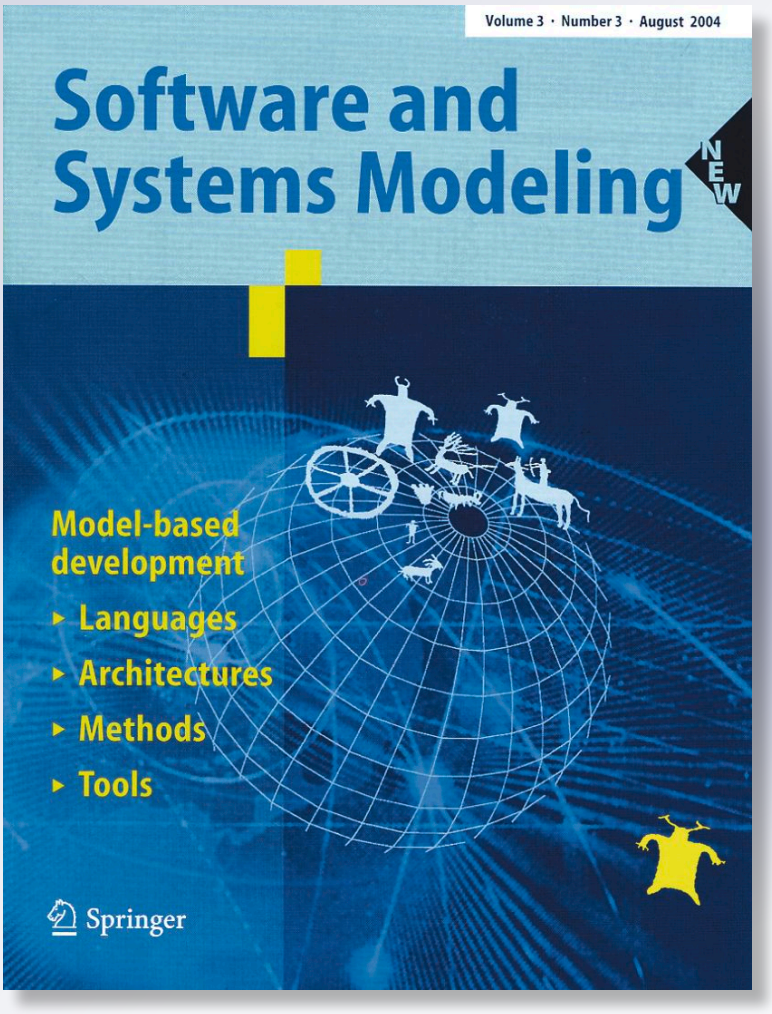
# *Approaches to modeling business processes: a critical analysis of BPMN, workflow patterns and YAWL*

**Egon Börger**

**Software & Systems Modeling**

ISSN 1619-1366
Volume 11
Number 3

Softw Syst Model (2012) 11:305-318
DOI 10.1007/s10270-011-0214-z




Springer

Softw Syst Model (2012) 11:305–318
DOI 10.1007/s10270-011-0214-z

EXPERT'S VOICE

# Approaches to modeling business processes: a critical analysis of BPMN, workflow patterns and YAWL

**Egon Börger**

Received: 1 August 2011 / Accepted: 22 August 2011 / Published online: 11 September 2011


**Abstract** We investigate three approaches describing models of business processes: the OMG standard BPMN in its recent version 2.0, the workflow patterns of the Workflow Pattern Initiative and their reference implementation YAWL. We show how the three approaches fail to provide practitioners with a suitable means precisely and faithfully to *capture* business scenarios and to *analyze*, *communicate* and *manage* the resulting models. On the positive side, we distill from the discussion six criteria which can help to recognize practical and reliable tool-supported business process description and modeling systems.

**Keywords** Business process modeling · BPMN · Workflow patterns · YAWL

## 1 Introduction and evaluation result

A recent book on business process automation [46] starts with the statement that after almost two decades of extensive research and development and despite numerous standardization efforts in the field (in particular XPDL, BPEL, BPMN, EPC, and UML Activity Diagrams), it remains the case that

> there is a lack of consensus about how business processes are best described for the purposes of analysis and subsequent automation (p. 5)

and that in particular

> Standardization efforts in the field have essentially failed (p. 5)

As fundamental reason for this problem the editors indicate that

> Process modeling languages tend to lack the concepts to be able to deal with the broad range of requirements one may encounter when trying to precisely capture business scenarios (p. 5)

making clear that in this evaluation they are

> not concerned with *expressive power* but with *suitability*, a notion that refers to the alignment between a modeling language and a problem domain. (p. 9)[1]

In order to solve the identified problem, the editors propose again their workflow patterns (WPs) in the book, together with an academic reference implementation.

In this paper, we explain why the recent version 2.0 [10] of the OMG standard BPMN deserves to appear in the list of failed standardization efforts in [46, p. 5] and also why the approach proposed once more as a remedy in [46] is a failure. More precisely the content of this paper is as follows:

### Evaluation results

– In Sect. 2, we survey various weak points in BPMN [10] which make the standard

Communicated by Bernhard Rumpe.

E. Börger (✉)
Dipartimento di Informatica, Università di Pisa, 56125 Pisa, Italy
e-mail: boerger@di.unipi.it

[1] In the introduction to [49] the authors describe suitability operationally as a 'notion of expressiveness which takes the modelling effort into account', i.e. an ergonomic pragmatic notion that considers the ease with which domain experts can express, communicate and handle real-life situations in terms of the given concepts.

- fail to guarantee that standard-conforming business process models are interoperable (platform-independent),
- fail to constitute a basis for reliable communication of business process models between different stakeholders,
- fail to be implementable with the claimed 'generality' without restricting numerous 'variation points' by imposing semantically significant restrictions of concepts which are left underspecified in the standard document.

– In Sects. 3 and 4, we explain on the background of this critical evaluation why both

  - the workflow patterns proposed and managed since 1999 by the two senior editors of [46] and their group through the vehicle of the Workflow Patterns Initiative [50] and
  - their reference implementation (called YAWL for 'Yet Another Workflow Language') to which the book [46] is devoted

  fail to provide the practitioner with (to use the editors' own words) 'suitable concepts' to guarantee that 'business processes are best described for the purposes of analysis and subsequent automation' and, in particular, contrary to the claim of the authors of WPs and YAWL, do not constitute an appropriate means for comparing and evaluating different business process modeling (BPM) systems.

This is not the place to propose a solution to the problem identified at the beginning of [46] and not solved by the book. Our goal here is to initiate a debate among the BPM experts about how to computationally support business process (BP) development and use in a manner which is *reliable in practice*, i.e. such that the BP technology put into place in an enterprise bridges the gap between

– what the system does when executing a BP,
– what the BP experts (who had to formulate the requirements for the system and to design it) and the IT technologists (who had to implement the system) thought it should do,
– what the different users think the system does

in a way that is controllably correct (correctness in the sense of the applications). As a methodological guideline, stepping back from the critical evaluation, which to a large extent is guided by the arguments put forward by the proponents of BPMN and WPs, in Sect. 5, we formulate a set of criteria for satisfactorily distinguishing reliable BPM systems from those which are not appropriate for the practitioner's daily work.

## 2 Problems of BPMN 2.0

In this section we explain why the OMG standard for BPMN (Business Process Model and Notation, previously called BPM Notation, see [10] for its most recent version 2.0) misses the main objective of a satisfactory standardization effort in the field, namely to guarantee that business process models conforming to the standard are interoperable (platform-independent) and constitute a precise basis for reliable communication between different stakeholders. Furthermore, the standard, as formulated in [10], appears not to be implementable due to the numerous behaviorally relevant issues it simply leaves open; this is the reason why implementations of BPMN concepts typically realize only subsets of the so-called standard and still are often only partially compatible with each other.

The shortcomings of the BPMN standard are well-known and have been observed (for different versions of the standard) by various authors, see for example [7,9,16,22,36,52,53,58] (the list is by no means exhaustive), so that we can restrict ourselves in this section to listing and briefly commenting upon some of the major points where the standard document fails.

– Numerous *ambiguities* in the descriptions and *underspecifications* of semantically relevant concepts pervade the standard document and leave space for incompatible (but, due to the lack of precision, standard 'conforming') interpretations in design, analysis and use of BPs. This is not a question of formal or informal descriptions[2] but simply a lack of precision, since a natural language, scientific, mathematical or pseudo-code description can be precise without being formalized.[3]
  The *lifecycle* concept is an example of an underspecified feature, particularly in relation to the equally underspecified *interruption* mechanisms like exceptions or cancellation or compensation for transactions. E.g. what about nested exceptions, especially in the case of multiple enabled nested interrupt events (say boundary timer

[2] [10, p. 425] *expressis verbis* did not adopt a 'formalization. . .using mathematical formalisms'.

[3] An experiment that illustrates well this point has been published in [19] where the contributing authors have been asked to produce a precise set of requirements for a small case study by first defining, in a language of their choice, a formal specification, which captures the given informal requirements and then retranslating this specification into natural language.

events): should one or all of them fire and in which order?[4] Or what is the cancellation scope for subprocesses or running instances of multiple instances within a process instance? What are the *completion* criteria for activities, for example in the case of simultaneous completion of multiple subprocess instances?
Another underspecification concerns *expression evaluation*. When should expressions (particularly event expressions) be evaluated? Depending on the type, it could (probably should) be either before or at process start, or upon state change or when a token becomes available. In case multiple events use the same non-deterministic expression definition, is the evaluation the same for each event and are all the events which listen to a common definition triggered simultaneously or is only one of them (and does this one consume the event or not)?[5] Also, the interaction of *transient and persistent triggers* remains undefined; it is semantically important to know, for example, when transient triggers become obsolete.

– BPMN provides only *poor conceptual support* for numerous features which are characteristic of the design and management of business processes.
  - A general notion of state is missing and, as a consequence, the specification of relevant *data* dependent conditions is only poorly supported. This makes the data management of an executable BPMN version compiler-dependent and not portable. For example, data objects, which are associated with activities or sequence flow, are used only informally, in particular with underspecified assumptions on the input/output selection at task nodes or sequence flows. Another example is data dependencies between different processes whose description is relegated in [10] to using informal associations instead of referring to shared locations.
    It is admittedly arguable, and this depends upon the level of abstraction of the targeted system view, which BP elements (e.g. attributes such as variables, named properties, or data in general) can remain graphically invisible without losing correct understanding at the chosen level of abstraction. The crucial criterion is how well the practitioner is supported when walking through the different levels of detail (refinement levels). One of BPMN's main shortcomings is here; see the discussion of refinement mechanisms below.[6]
  - Management of *resources* can be expressed only via lanes (actors, roles, etc.) or performers of user or manual tasks.
  - The standard does not support process *structure* at the risk of producing incomprehensible spaghetti diagrams (see [38]). To let the modeler adhere to some useful structuring discipline one could have restricted the class of admissible BPMN diagrams to the reducible flow graphs, a classical concept in compiler construction for the compilation of structured programs [1]. More generally, there is only poor support for BP (de-)composition methods which trigger *modular component-based design*.
  - *Communication* and *process interaction* are poorly supported for concurrent execution, e.g. of independent (not embedded) subprocesses or of processes belonging to different parts of one organization or to different cooperating organizations.

– BPMN comes with a *plethora of interdefinable constructs*. Instead of defining a core of independent constructs in terms of which other constructs can be defined, as suggested in [9] for a previous version of the standard (it was also suggested to the standardization committee). The fuzzy overlapping of different constructs prevents 'closed' descriptions of individual constructs in one place and makes their comprehension unnecessarily complex by forcing the reader to simultaneously and repeatedly consider multiple sections of the standard document. It also creates the problem that where the definitions overlap they have to be consistent; this problem is not considered in the standard document. Furthermore, a statistical evaluation (of BPMN 1.1) shows that 'the average BPMN model uses <20% of the available vocabulary' and that, out of the more than 50 graphical elements in BPMN, 'Only five elements (normal flow, task, end event, start event, and pool) were used in more than 50% of the models we analyzed' [60,61].
– The BPMN standard claims to (and indeed should) support the definition of models which serve the three major purposes of business process descriptions, namely model *design and analysis* (requiring accurate conceptual models in particular for high-level development and management support), model *implementation*, where the models play the role of the specification of software requirements

[4] We mention *en passant* an unfortunate omission, namely that the standard foresees no predefined interpreter exception types.

[5] A comparison with language definitions in programming should clarify our point: In C for example a construct may have undefined, unspecified or implementation-dependent behaviour. Undefined behaviour permits any behavior (e.g. null-pointer dereference or out-of-index-bound access to arrays). Unspecified behavior refers to a set of possible well-defined behaviors (e.g. the evaluation order for expressions), while implementation-dependent behaviour means that the compiler has to provide a definition (which must be documented). In this way, the programmer is made aware of the possibility of different behaviors of a construct and its related problems, whereas the BPMN standardization does not do this.

[6] A related and not to be neglected issue is that an increase in graphical notation yields an increased complexity in the meta-model and makes transparent faithful implementation more and more impractical.

and are transformed into executable models, and the *use* of models (user model for process execution, monitoring and management).[7] The standard document fails to provide a seamless systematic mechanism for refinement from conceptual to executable models, which is necessary to guarantee the reliability of the implementation.[8] As a consequence, there remains an *unmediated gap between conceptual and executable BPMN models*: executable BPMN models (in particular if obtained through compilation to more detailed languages like BPEL or to code) are hard to grasp and analyze for human readers [37,54]; there are no standardized view mechanisms supporting the extraction of abstract (e.g. user or management) models from detailed ones (e.g. software models), as is the case with, for example, the Workflow Communication Modeling Language WF-CML [59] in its definition of communication-specific abstractions of workflows concepts. An analogous problem results from the poor interweaving of different BPMN diagram types, in particular that no consistency criteria are imposed for them, for example collaboration versus choreography versus process diagrams. The practical relevance of this problem for the use of BPMN is again evidenced by a statistical evaluation which 'found anecdotal evidence that two groups of modelers use BPMN... one group uses BPMN to specify inter-organizational settings (process choreography)...The other BPMN user group is leaning more towards workflow engineering (process orchestration)' [60,61].

In conclusion, one can say that while missing conceptual streamlining makes the standard document unnecessarily complex, more importantly, ambiguities, underspecifications and missing support for semantically relevant concepts in the BPMN standard make the document miss the standardization target. In fact, within the boundary of what is called 'standard-conforming' it may happen that:

- one and the same construct is understood differently by different stakeholders, at the risk of making communication between them unreliable where it uses the construct,
- one and the same construct is compiled by different compilers to semantically different execution behaviors, the reason for this being that to define the compilation method for each unspecified, but behaviorally relevant feature, a decision about its interpretation has to be taken. Since these decisions remain invisible in the conceptual or user model, the user may understand the described process in a way that is different from that which happens during its execution. Furthermore, since the decisions taken for different compilers may be incompatible, the execution semantics of a BPMN model remains compiler-dependent so that the model may not be interoperable and maybe not even portable [20,33,34,37,54].

The way out of this which is advocated by some experts is to provide a 'reference implementation' where all omissions in the standard document are clarified. But then the reference implementation (assuming everybody agrees on just one) would become the true standard,[9] which contradicts the overall standardization goal. BPMN models would have to follow new releases of the reference implementation instead of the other way round.

## 3 Problems of WPI workflow patterns

In 1999, the two senior editors of [46] started the *Workflow Patterns Initiative* at the Universities of Eindhoven and Queensland (WPI [50]) and reproposed the WPI workflow patterns in [46] with the declared goal

> to describe process modeling requirements in an implementation independent manner [46, p. 5] and
> to distill the essential features of the many workflow management systems that existed. [46, p. 9]

They express the hope that as a result,

> This would allow an unbiased comparison of different approaches to the specification of executable business processes and provide the basis for the adaptation and refinement of existing techniques as well as supporting the development of new approaches. [46, p. 9]

In this section, we explain the reasons why we consider the WPI workflow patterns to be

[7] See also the statement by the convenor of the standardization group at http://www.workflowpatterns.com/vendors/bpmn.php: 'The primary goal of BPMN is to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes.'

[8] A precise refinement concept could have been defined on the basis of a rigorous modeling framework, without need to push it as far as a 'formalization...using mathematical formalisms' [10, p. 425], and could have been used for mediating between the different granularities of high-level and more detailed views, e.g. corresponding to the three BPMN levels (*descriptive*, *analytical* and *executable*) considered in [43].

[9] The analogy in programming is to accept a language compiler and its embedding into a machine environment as defining the semantics of programs of that language.

– neither a 'suitable'[10] way to 'describe process modeling requirements,'
– nor a 'suitable' measure to evaluate BPM approaches.

We show that they do not 'distill the essential features' of BPM systems, come without a rational foundation. We also show that they have their own strong bias (in particular to quantity, neglecting conceptual simplicity and modularity).

### 3.1 WPI workflow patterns are not well founded

The workflow patterns, as presented by the WPI [50], come without pragmatic or rational foundation. In fact, there is *no statistical underpinning* showing how frequently which patterns appear in real-life business processes. Experimental data like that produced for BPMN constructs in [61] should have been put in place to validate the pattern selection. But, in addition, *no conceptual justification* is provided for the choice[11] or the classification of patterns proposed in the original sources [40,51].[12] Instead an exponential growth can be observed starting with 20 workflow patterns in 2003 [51] going through 43 WPs in 2006 [40] and reaching a praised 126 patterns (obtained by adding patterns for the so-called data and resource perspectives) in 2010 [46]. No reason is given that this growth has a *fundamentum in re* and a limit.

This apparent bias of the WPI towards quantity to the detriment of conceptual simplicity, succinctness and consideration of (de-)composition techniques has a methodological drawback with severe practical consequences. In fact, most patterns are not of fundamental character but are easily definable from a small set of more basic and rather simple patterns. Shortly after the appearance of the 43 patterns, it was shown in [6] how they can all be rigorously defined by *instantiations of 8 natural, generic schemes* where the instantiations are instantiations of the parameters of precisely-defined abstract models. Furthermore, these eight schemes represent simple, well-known, basic sequential or concurrent programming constructs[13], appropriately parameterized to capture the targeted class of 43 workflow patterns.

A similar conclusion was made independently in 2009 by a team of students at KIT, who, in their bachelor thesis [21], modeled the 43 patterns by a mapping from 25 more basic (appropriately generalized) 'choreography patterns', using the S-BPM tool [31]. This represents a quantitative reduction by 42% and it is stated that it could be further improved, using the S-BPM language, to fewer than 18 basic patterns.

One can conclude that the WPI-patterns, whether 20 or 43 or 126,[14] contrary to what is claimed by the WPI group, do not 'distill the essential features of the many workflow management systems' [46, p. 9]. They resemble very much molecules rather than fundamental elements. What is useful for the working business process expert is not an unlimited variety of 'molecules' but a small table of truly essential, simple and clearly defined BP 'elements,' which allow one to produce in place (or via some preconfigured modular definitional scheme) whatever combination is needed[15] for each specific process or group of processes to be developed to satisfy given requirements. The WPI WPs, on the contrary, come without any concept of (de-)composition.

### 3.2 WPI workflow patterns are badly described

The workflow patterns as presented by the WPI [50], again contrary to the claims of the WPI group, are not a 'suitable' way to 'describe process modeling requirements in an implementation independent manner' [46, p. 5]. In fact, they come in [40,51] with descriptions in natural language, which, in many places, are ambiguous[16] or incomplete (leaving behaviorally relevant features unspecified) and sometimes overspecified by implementation features (mostly belonging to coloured Petri nets). They share this fuzzy character of their semantics with BPMN and, for this reason, they cannot serve as a satisfactory BPM evaluation benchmark and cannot serve as a substitute for a BPM standard.

The formalization in [32], for all practical purposes, does not really change the situation. It uses an *ad hoc* defined graphical Workflow Pattern Specification Language called WPSL; despite the length of the paper, the formalization is admittedly incomplete[17]. Due to its complexity, it can hardly

[10] Suitability is taken in this paper in the editors' sense as 'a notion that refers to the alignment between a modeling language and a problem domain' [46, p. 9]

[11] In [32, p. 1] it is even conceded (referring to [51]) that 'the selection of these patterns was done in an ad-hoc manner'.

[12] *Nomen est omen*: for some pattern even the name is badly chosen. E.g. in the 'interleaved parallel routing' pattern 'the activity instances do not interleave, and they are not executed in parallel' [56, p. 146] so that in op.cit. the pattern is renamed to 'sequential execution without design time knowledge'.

[13] Concerning related aspects of the close connection between program and business process development, it may be of interest to the reader to consult [23].

[14] In view of the rational reconstruction of what were, at the time, 43 workflow patterns in [6], and a similar result for communication and interaction patterns in [3], we have good reason to expect that the actual set of 126 patterns could also be reduced to a small number of appropriately parameterized schemes.

[15] A small set of easy-to-grasp patterns which covers a large percentage of every-day applications is better than a comprehensive set of more and more complex, and thus more and more difficult to understand, patterns one can hardly remember.

[16] The 'rather ambiguous' character of these descriptions in natural language is conceded in [32, p. 1] as a motivation for a formalization of the patterns; see the discussion in the next paragraph.

[17] 'The current specification covers only the control-flow perspective and does not consider the data and organizational aspects. . .. The scope

be claimed to be a proposal for a working definition to business process practitioners. Furthermore, it does not satisfy the criterion the editors put forward in [46, pp. 14–15], namely to be 'generally accepted'[18]; we found no reuse of the formalization after 2006, neither by its authors nor by others.

The precise definition for the intended behavior of WPI workflow patterns via the reference implementation YAWL proposed in [46] is, by this very definition implementation dependent, and, as a consequence, of no use if the goal is to 'describe process modeling requirements in an implementation independent manner' [46, p. 5]. But various problems remain even in the case in which a definition in terms of a reference implementation is accepted. These problems are directly related to YAWL and are discussed in Sect. 4.

### 3.3 WPI workflow pattern are no suitable BPM benchmark

In this section we show that contrary to what is claimed in [46, p. 9], the WPI workflow patterns do not represent 'suitable' benchmarks to establish 'an unbiased comparison of different approaches to the specification of executable business processes'.

The 'vendors corner' of [50], which we consulted in May 2011, declares

> that the evaluation of workflow products on this site only considers direct support,[19] i.e.
>
> > Is there a feature in the system/language, directly addressing the pattern?
>
> Clearly, the absence of direct support does not mean that the pattern cannot be supported at all. In some cases, there may be an easy intuitive way to *work around the problem*[20]. In other cases, it is impossible to work around the problem and the designer has to resort to developing additional software.

This *proviso* suggests pattern featurism as a quality criterion for BPM systems and tacitly assumes that missing 'direct support' represents a problem. But this assumption and the restriction to 'direct support' fundamentally misjudge definability as equivalent to '*work around the problem*'. More than 2000 years of scientific practice have established definability as a royal road to stepwise problem solution, modularity and reduction of complexity, and by no means as an expedient to 'work around the problem'. 'Suitable' BPM feature support will come as a support for a small set of well-founded core modeling constructs, together with a rigorously defined support for modular composition and definitional extensions by (possibly hierarchically structured) layers of additional features. The latter may be introduced to reduce the modeling effort, but also, for example, because they are of less frequent use than the core constructs or needed only for certain groups of users. Who would consider as *work around the problem*, when model libraries are defined for a given BPM system to extend, what is made available 'directly' to the practitioner?

The editors themselves in [46, p. 14] present as an advantage to having 'a minimal set of constructs, rather than a construct-per-pattern approach' in the 'reference implementation' (YAWL) of their workflow patterns, though this fact implies that—contradicting the direct support principle they advocate—some workflow patterns have a definition from other constructs in YAWL instead of being directly supported.

*En passant* let us mention a surprise that shows up when YAWL, which is proposed as 'reference implementation' for the patterns, is benchmarked along the pattern support criterion: for some patterns the proposed definition in YAWL seems not to work the way one would expect from the (natural language) specification. In [56, p. 196], it is shown that the proposed YAWL simulation of the *discriminator* pattern (namely by combining an exclusive OR join with a cancellation region) 'exhibits significant semantic differences with the specification of the discriminator' and that 'the *N-out-of-M join* is not completely expressed' (via multiple instances) [56, p. 197], namely not for the case where concurrent process branches belonging to different activities have to be synchronized. So—unless one takes the pattern implementation as the pattern definition—YAWL itself does not reach completely the WPI benchmark?

A serious pragmatic issue which sheds doubts on the appropriateness of the WPI workflow patterns as a benchmark for BPM systems is that the following three questions remain unanswered:

- How often which workflow patterns need support, whether direct or definitional, in real-life applications?
- How representative are the 13 tools (out of over 170 worldwide[21], including the BPMN approach!) which

Footnote 17 continued
of the language is limited to a single case, covering the details of the case routing, while leaving the external relationships with other cases, processes, and external environments out of consideration' [32, p. 5].

[18] See Sect. 4.3 for a discussion of this criterion.

[19] A colleague pointed out that there are examples where also some form of 'workaround' is accepted by the WPI for the 'certification' that a pattern is supported, e.g. the evaluation of the *Cancelling Discriminator Pattern* as supported by BPMN, although its definition in terms of BPMN constructs is rather involved. When does the WPI decide that a 'workaround' is accepted and on the basis of which (hopefully scientific) criteria is such a judgement made?

[20] Our emphasis.

[21] Figure given by Janelle Hill, Research Vice-President at Gartner, as reported in http://www.computerweekly.com/Articles/2007/04/03/222851/BPM-is-the-next-39killer-application39-says-Gartner.htm, consulted by us on July 13, 2011.

appeared in May 2011 as evaluated at the WPI Vendors Corner 'in terms of the original control-flow patterns'?
– What guarantee is offered that these evaluations are unbiased, in particular, free of any 'vested interest' of the group behind WPI?

As a conclusion of this section, one can say that the workflow patterns as defined by the WPI group in [50] do not appear to be 'suitable' for satisfactory business process model descriptions or BPM system evaluations. This does not exclude that a (why not standardized?) well founded accurate definition of (a) truly fundamental workflow patterns and (b) practical ways to extend and compose them (why not into *Business Process Lines* (BPLs), adapting well-known techniques from Software Product Lines [35]?[22]) could provide appropriate means for satisfactory BP model descriptions and BPM system evaluations (see the description of the modularization criterion in Sect. 5).

## 4 Problems of YAWL

In this section, we also explain the reasons why YAWL, presented in [46, p. 11] as 'a reference implementation that supports the Workflow Patterns' of the WPI, fails to provide the business process practitioner with 'suitable concepts' whereby 'business processes are best described for the purposes of analysis and subsequent automation' [46, p. 5]. The analysis is not intended as an evaluation of YAWL's technical features[23] which is more appropriately performed by the users of the system. Instead, it is guided by the claims made for YAWL in [46]. This leads us in particular to analyze in Sect. 4.2 the semantical foundation of YAWL using Petri nets, an investigation which reveals the weakness of Petri nets as a modeling and analysis framework for complex business processes.

### 4.1 WPI workflow patterns versus YAWL: a vicious circle?

In [46, p. 14], it is claimed that YAWL, as a reference implementation to support the WPI workflow patterns, has been

> developed without the pressures of vested interest and a sole focus on providing *comprehensive support for the Workflow Patterns*

*En passant*, let us observe that the way the WPI group propagates the set of patterns (that has been growing since 1999) makes the support for these patterns by a reference implementation appear, in fact, as an obvious (not vested) group interest.

More importantly, one can observe a vicious circle appear in the respective justifications of the WPI workflow patterns and YAWL: on the one hand, YAWL is motivated and justified as a support for the WPI workflow patterns; while, on the other hand, life (read: their existence as schemes with rigorous behavior) is breathed into these patterns only by their definitional support in YAWL—and this is not surprising since they have no 'natural', formalism-independent, precise semantical foundation of their own, as has been pointed out in Sects. 3.2 and 3.1. Certainly, the semantic definition of the WPI patterns by a reference implementation does not satisfy the criterion of 'implementation independence', which however is explicitly required in [46, p. 5] for 'suitable' approaches to process model descriptions.

However, if one realizes that the informal description of the WPI patterns and the definition of their behavior in YAWL are not independent of each other but only represent two sides (requirements and their implementation) of the same proposal, in a way different from the BP practitioner, a software engineer could be inclined to accept that the rigorous definition of the pattern behavior is provided within YAWL—*if* the semantic foundation of YAWL is 'suitable' for BPM. We investigate this question in the following section.

### 4.2 Problems of the semantical foundation of YAWL

In this section, we show that the purported semantic foundation of YAWL using coloured Petri nets is not 'suitable' for the practice of BPM. The editors of [46] state the following:

> In YAWL, Petri nets were taken as a starting point and extended with dedicated constructs to deal with patterns that Petri nets have difficulty expressing, in particular patterns dealing with cancellation, synchronization of active branches only, and multiple concurrently executing instances of the same task. (p. 10)

> . . .the semantics has been defined in terms of a large Colored Petri net (pp. 14–15)

The following reasons are given for the choice of Petri nets and their various extensions (in particular workflow nets and reset nets) as a basis for the semantical foundation of YAWL [46, p. 10] (our emphasis):

> three reasons why Petri nets would make a good candidate. . .for workflow specification. . .
>
> – the *graphical nature* of Petri nets,

[22] A product line is a family of similar programs. Each program is constructed by a . . . combination of features, where a feature or feature module encapsulates program fragments. . . (Quoted from the abstract of D. Batory's talk 'Towards Verification of Product Lines' at the Interactive Theorem Proving (ITP 2011) Conference, Nijmegen 2011).

[23] Some of the problems we identified for BPMN in Sect. 2 are encountered also in YAWL, for example the poor support for communication and interaction between independent subprocesses.

– their *explicit representation of the notion of state*, and
– the existence of *analysis techniques*

We show in the following three subsections that none of these reasons is convincing, either from the scientific or from the practical point of view. The discussion proceeds in the order of importance of the three features.

**Insufficient notion of Petri net state and computation step** Neither the notion of state of (possibly coloured) Petri nets, nor the computational power of Petri net transitions are general enough to deal 'suitably' (case 1) or at all (case 2) with complex business process objects or transitions.

– **Case 1** Concerning suitability, typically encodings are needed for the representation of complex BP objects or operations, contradicting the request in [46, p. 9] that the concepts of the modeling language should be aligned with the problem domain.
Due to space limitations we can illustrate this point here only for one characteristic example, which stands for many others. Reset nets and YAWL provide an encoding based on tokens and special arcs of the abstract concept of process deactivation (interestingly enough called 'cancellation'), a concept which, however, involves no tokens but only the notions of process deactivation (an operation) and of to-be-stopped processes (a typically parameterized predicate). In these terms, the concept can be abstractly described as follows (avoiding the introduction of additional arcs which with growing net size quickly obfuscate the diagram structure and avoiding the problem related to the token removal function explained in [56, pp. 199–200]):

**forall** $p \in ToBeStopped(params)$ DEACTIVATE($p$)

The unavoidable impractical complexity of various Petri-net based YAWL definitions for the semantics of high-level BP concepts is due to the missing coding-free support of data types as they typically appear at the BPM intersection of business and information technology. Examples are multiple instance tasks with upper and/or lower runnable instance bounds, completion threshold, the dynamic/static start attribute, various resource allocation strategies, etc. All these concepts can be expressed directly, avoiding extraneous token-based encodings.
– **Case 2** Concerning feasibility, the case where, due to their restricted notion of state and transition step, Petri nets cannot describe certain operations which are typical for the BP domain, the problem with the Petri net model of computation becomes evident when one has to describe non-local or truly concurrent or recursive business process behavior[24]; well-known examples are the OR-split and OR-join [11], 'synchronization of active branches only, and multiple concurrently executing instances of the same task' [46, p. 10], a priori unbounded loops in workflows [55], etc. Similar problems have been encountered when trying to map BPMN diagrams [14,15], [22, Sect. 4.3.3] or UML 2.0 Activity Diagrams [44] to Petri nets.
Furthermore, the non-deterministic execution model of Petri nets hides an implicit 1-core interleaving assumption ('no 2 transitions fire simultaneously') which is inappropriate to deal with real-life concurrency of business processes and their multicore machine implementations. In this respect, even BPMN is better because it offers at least the concept of pools to deal (though in a rudimentary way) with inter-organizational BP aspects.

As a consequence further additions to Petri, workflow and reset nets are needed to provide the means to describe BP features which are not covered by the restricted Petri-net model of computation. An example in YAWL is the classification of *task start* depending on whether the trigger is automatic, provided by the user, external or based on time. Another example in YAWL is the mechanism introduced to represent a possibly positive execution time of a transition, instead of the 0-time execution of Petri net transitions, namely by tokens which reside at the transition during the execution. Another example is the so-called 'open workflow net' extension (which uses strong fairness assumptions) defined in [27] to include communication via asynchronous message passing. Since in real-life business processes synchronous communication also occurs, yet another extension of open workflow nets is needed to handle it too?[25]

As a conclusion of this subsection, one can say that the statement in [46, p. 5] about languages which 'lack the concepts to be able to deal with the broad range of requirements one may encounter when trying to precisely capture business scenarios' directly applies to Petri nets, workflow nets, reset nets and other extensions proposed for modeling business processes.

**Existing Petri net analysis tools applicable?** It remains to analyze how 'suitable' the praised tool-supported, Petri-net based analysis techniques are for real-life (not academic examples of) business processes.

[24] In [29] Petri nets are characterized as process rewrite systems with parallelism, a strict subclass of more general process rewrite systems with parallelism and sequential composition, formally Petri $= (P; P) \not\subseteq (P, S; P, S)$.

[25] In fact in controller design so-called net condition/event systems are used to model the synchronous firing of multiple transitions, see [24].

One question is how many and which of these techniques have been ported from (usually subsets of) Petri nets to the necessary YAWL extensions. To mention just one example, Petri-net tools usually assume (for fairness reasons) that loops terminate; this excludes possible execution paths in relevant real-life BP behavior [55], which is therefore only incompletely covered by analysis tools working under that assumption.

Another question is whether for high-level BPM analysis, the focus on control flow (read: the almost complete abstraction from communication, data and resources)[26] or on fairness or liveness or similar properties traditionally analyzed for Petri nets is appropriate. For example, liveness analysis is irrelevant in the presence of timeouts and user interrupts. Similarly, fairness is mostly studied as a property of infinite runs (which, in practice, do not occur); for finitary real-life interpretations, it is typically handled by the scheduler.

It comes as a surprise that a tool-supported language like YAWL, proposed by a group of academic researchers with property verification as explicit target, only cares about checking standard properties (of limited practical value[27]) by (hopefully applicable) existing tools instead of including into the language at least the possibility to express some form of pre/post condition or invariant property a BP model is intended to satisfy (and why not introduce the possibility into the tool to perform such checks). Such a feature (which, for BPM, would be innovative) would not only simplify BP model validation (testing) and verification (property proving) efforts (see for example [2,53]), but would make them available for application-specific properties of interest to the user or process manager. Why not learn from software engineering and implement a *Business Process Lines* approach which links feature-based modeling to stepwise validation and verification the way it is done for Software Product Lines (see [4,12,25,47])?

**Minor role of graphical nature of Petri nets** It comes as a surprise to see the graphical nature argument brought forward in favor of using Petri nets for BPM.

First of all, the graphical nature of Petri nets contributes only in a minor way to the graphical elements which are typically used for BPM languages, as one can observe in the graphical notation introduced for numerous constructs in YAWL (though their number does not equal the more than 100 graphical elements proposed by BPMN 2.0).

Furthermore, what is graphically displayed is the token/transition determined character of Petri nets which, as observed above, often unnecessarily complicates process representations by low-level details (token-focussed workflow control). Those who are old enough to have studied Petri nets shortly after Petri published them in his doctoral dissertation will remember the fascination this new computation model exercised, but also the disillusion that was triggered by the huge, wall-covering, unreadable and scarcely controllable diagrams which resulted from attempts to model complex algorithmic or real-life industrial processes by such nets.[28] We invite younger readers who want to experience a concrete technical comparison of the ergonomic quality ('suitability') of Petri nets with respect to models in other modeling approaches to look at two sets of models for a small number of well-known network algorithms, namely the relatively complex and not easy to grasp Petri net models in [39] and the simple, succinct, abstract companion models in [8, Ch. 6.1].

As a conclusion of Sect. 4.2, one can say that although there is no doubt that, given the mathematical character of Petri nets, it is not impossible to provide via appropriate extensions a mathematical foundation of YAWL, nevertheless the reasons put forward in [46] do not convince that this approach to a foundation for BPM is 'suitable' for the working BPM practitioner. There remains the question of how YAWL is positioned as a tool environment among its competitors. This issue is discussed in the next Sect. 4.3.

### 4.3 Dismantling the claimed 'Unique Position' of YAWL

In [46, pp. 14–16], it is claimed that 'YAWL has a number of features that position it uniquely in the crowded[29] field of BPM'. We explain in this section why YAWL does not satisfy any of the following three major criteria among those which are brought forward to support this uniqueness claim (our numbering and emphasis):

1. a *minimal set of constructs*, rather than a construct-per-pattern approach
2. sophisticated *flexibility support*. . .(given that) exceptions may arise. . .processes may evolve over

[26] According to [45, p. 415], traditional flow-driven assembly and orchestration of service components are 'too rigid and static to accommodate complex, real-world business processes'.

[27] This is not to argue against applications of Petri-net tools to appropriate problems, as for example in [48]; there, the absence of deadlocks and synchronization failures is verified for certain workflow graphs by translating the latter to a subclass of Petri nets which is tailored towards workflow analysis and equipped with a tool to verify such properties. Proceeding this way works well if the translation from the application domain models to Petri nets (and back) are (proven to be) correct with respect to the target analysis algorithms and do not force the domain expert to understand the details of the Petri net models.

[28] We are here only concerned with the 'suitability' of the computational model of Petri nets for the practice of BPM and definitely do not want to imply any negative judgement on the mathematical beauty of (some of) Petri net theory [57] and its relation to the theory of Gröbner bases [26,28].

[29] It is inappropriate for a book which wants to be considered as a scientific text to use so many technically meaningless, emotional or marketing terms like 'crowded' in this sentence.

time due to changes in a business and/or its environment
3. a *formal foundation*, that is, both a precisely defined syntax and a precisely defined semantics.

To conclude the uniqueness-claim discussion, we point to two examples of (not academic) BPM systems which satisfy all three of these criteria—a counterexample to the YAWL-uniqueness claim—respectively other criteria which from the practical point of view are not less interesting.

**Minimality of constructs** The minimality is in the eye of the beholder and claimed but not proved.[30] Furthermore, as explained in Sect. 3.3, the minimality property claimed for YAWL implies that YAWL does not satisfy the advocated criterion of 'direct' pattern support as a measure for qualified BPM systems.

In addition, the minimality property should hold, if at all, then, for workflow patterns, as explained in Sect. 3.1. Why should it hold for their implementation?[31]

Certainly a small set of constructs does not position YAWL uniquely with respect to other BPM systems. For a counterexample see below.

Above all, though, the minimality criterion itself is questionable as a BPM quality measure. The problem is not the quantity but the conceptual and pragmatic quality ('suitability') of features offered to the modeler in the form of basic constructs and of easy to use definitional schemes (to compose specific features out of the givens) or of library constructs (see the discussion on the direct support principle in Sect. 3.3).

**Flexibility** Also the two features mentioned to illustrate the flexibility claim for YAWL, namely exception handling and process extensions to reflect the evolution of requirements, do not position YAWL uniquely. There are other BPM systems which support such flexibility. For a counterexample see below. In reality, flexibility should come as a result of general support for abstraction and modularity based upon a practical decomposition theory. In this respect YAWL does not seem to stand out.

**Formal foundation** Concerning the formal foundation as one of the features which are claimed to position YAWL uniquely, it is stated in [46, pp. 14–15]:

> . . .the semantics has been defined in terms of a large Colored Petri net (CPN), which can interpret YAWL specifications. . .While it is sometimes claimed that certain standards or oft-used approaches have a formal foundation, the problem is usually that
>
> – either (1) the connection between the language and the formal theory remains unclear
> – or (2) the formalization is not generally accepted, and certainly not by a standard body.

Applying these two criteria to the language and the theory proposed in [46] results in the observation that the problem identified by these two criteria shows up right away for the formal foundation of YAWL. In fact:

1. The 'connection between the language and the formal theory' which is presented in [46] 'remains unclear'.[32] Despite the size of the book (676 pages), which is entirely devoted to YAWL, the 'large Colored Petri net' claimed to define the semantics of YAWL does not appear. Is it one of these large nets which by mere size are hard to grasp for humans and therefore not proposable?[33] More importantly the question remains how such a 'large' (how complex?) coloured Petri net can exist for the entire YAWL system (not just a subset) and how—should it exist[34]—can it help the practitioner to understand the semantics of YAWL? A precise semantics model is not there to be 'believed in' must be consultable by the BP user to answer questions about the language.
2. It seems not to be the case that (coloured) Petri nets are 'generally accepted'—outside the Petri community and at least comprising the BPM stakeholders—or accepted by a BPM standard body.

**Counterexamples** An example for a BPM system, which, contrary to YAWL, does satisfy the three 'uniqueness' features discussed in this section is the S-BPM system [31], which we discovered half a year ago (see also its recent analysis by Gartner [30]). It comes with a few fundamental BP-specific constructs (which are equipped with a tool-independent and epistemologically well-founded rationale), with two

[30] It is true that at the level of the YAWL editor, one sees only three kinds of split (AND, OR, XOR) with corresponding joins, multiple instance tasks and refinement. But much more shows up—and necessarily so to meet the practicality claim for YAWL—if one digs a little bit further into the system.

[31] A small number of constructs in the source and target system certainly simplifies and facilitates correctness proofs of an implementation (here in YAWL) of the source system (here workflow patterns). But [46] does not pursue such a system verification goal.

[32] A theoretically knowledgeable academic YAWL user with considerable industrial experience to whose criticism we submitted this statement—fully confirmed it.

[33] In [49] a formal semantics is defined for what is called 'the control-flow perspective' (not including the 'data and resource perspective') of YAWL; footnote 1 in Sect. 4.1 of that paper says that 'YAWL can be mapped onto high-level Petri nets. However, this mapping is far from trivial. . .we would like to emphasise that the semantics of YAWL is independent from high-level Petri nets.'

[34] It appears to us that the execution semantics of YAWL as a matter of fact is not defined by Petri nets but by more general state transition systems.

modular model extension disciplines (covering in particular exception handling and evolutionary changes) and with a simple precise model of its behavior that BP stakeholders can understand.[35]

An example of a system which comes with other features of interest to the practitioner than those mentioned to 'position YAWL uniquely' is the BPM tool developed by Signavio. It comes with free use for teaching and research (offered through the BPM Academic Initiative [42]) and permits modeling in different languages, integrates different workflow engines offered by other providers and various analysis tools and provides an open source engine with native support for a subset of BPMN 2.0 (lightweight Java approach).

This leads to the question discussed in the next section about significant methodological criteria to guide meaningful evaluations of BPM tools.

## 5 Evaluation criteria for BPM systems and a challenge

The critical evaluation of BPMN, WPs and YAWL in the preceding sections followed the arguments put forward by the proponents of these approaches so that our judgement comes in terms of the proponents' declared criteria. We could have conducted the investigation along the following six fundamental properties which characterize a good approach to system modeling (design and analysis) and are well-known from software development (*mutatis mutandis*).

A BPM system must serve the three major purposes of business process descriptions, namely *design and analysis* of models (requiring conceptual models, in particular, for high-level development and management support), *implementation* of models, where the conceptual models play the role of the specification of software requirements and are transformed into executable models, and the *use* of models (user model for process execution, monitoring and management). Correspondingly, a valuable BPM system can be recognized by the extent to which and the degree of practicality in which it realizes the following three fidelity and effectiveness features:

**Ground model support** *Support for correct development and understanding* by humans of models and their relation to the application view of the to-be-modeled BP, which is informally described by the process requirements. It is crucial to support such an understanding *for both model design and use* because these models[36] serve for the communication between (a) the BP experts who have to explain the real-world BP that is to be implemented, (b) the IT experts who need an unambiguous specification of the coding goal, and (c) the BP users who apply or manage the implemented process and need to understand, for their interaction with the system, that their process view corresponds to what the code does. This human-centered pragmatic property is the most critical one for BPM systems.

**Refinement support** *Support for faithful implementations* of models via systematic, controlled refinements (controlled either by experimental validation and/or mathematical verification). This IT technology-centered property is, methodologically speaking, the simpler one to achieve since it deals with precise entities (mathematical models) and can be dealt with using the enormous wealth of established refinement, transformation and compilation methods—if the first feature supports the construction of reliable (precise, correct, complete and minimal) ground models from which the implementation can start.

**Management support** *Support for effective management* (including monitoring, evaluation and evolutionary change) of models. This third property is again a critical one for BPM systems, and is not easily achieved since it involves the interaction between machines and humans who have to interpret and evaluate machine executions for BP (ground or refined) models, again bringing in conceptual (ground model and refinement) concerns when it comes to adapt the system to evolutionary changes.

To possess these three features, a valuable BPM system has to support two fundamental, well-known development and description methods (which are listed in the *Design* section of *Great Principles Category Narratives* in [13] under simplicity, one of the five 'driving concerns of software design and use' to 'overcome the apparent complexity of applications'):

**Abstraction** to support the practitioner in two respects:

- in the daily challenge to develop abstract models (ground models together with some refinements) out of concrete, real-life problem situations. This implies, in particular, the availability in the modeling language of a rich enough set of abstract data types (sets of objects with operations defined on them) to use so that one can
  - express the application-domain phenomena to be modeled (objects and actions) at the conceptual level without the detour of language-dependent encodings;

[35] Clearly this is not the place further to dwell on the properties of S-BPM with respect to BPMN, WPs and YAWL; but according to scientific practice we have to discharge our technical claim so that we point the reader to chapter 12 and the appendix of [17] where the three criteria can be checked to be satisfied by S-BPM.

[36] In the semiconductor industry they are called *golden models* [41], in [8] *ground models*. For an analysis of their characteristic properties and of their role for system validation and verification see [5].

  - refine the abstractions in a controlled manner by more detailed operations on more specific data structures.
- in coherent definitions of different system views (control-flow view, data flow view, communication view, view of the actors, etc.).

**Modularization** providing rigorous, abstract, behavioral interfaces to support structured system compositions into easy-to-change (possibly hierarchically organized) components. For BPM, it is particularly important that *modeling-for-change* is supported at all three major stakeholders levels: at the **Ground Model** and **Management** support levels because BP users and managers drive the evolutionary adaptation of BP models, at the **Refinement** support level because the high-level model changes have to be propagated (read: compiled) faithfully to the implementing code.[37]

Clearly for a BPM tool environment to satisfy these five criteria a sixth criterion is mandatory, namely that the system provides a

**Practical foundation** a *precise foundation a practitioner can work with*, i.e. understand and rely upon when questions come up about the behavioral meaning of constructs used by the tool.

Only such an accurate semantical foundation makes the other five criteria meaningful so that it would deserve position one among the six criteria, though in practice this is rarely recognized.

**A challenge** A technology transfer from the theory and practice of (feature-based) Software Product Lines to BPM could resolve the deplorable status of BPM standardization in three steps:

- identify a common kernel of major BPM tools in terms of control, action and communication features,
- build a precise high-level behavioral model for this kernel which reflects the kernel concepts (feature atoms and feature composition schemes) directly, avoiding encoding into any form of a priori determined description language other than the mathematical language of computing,
- model concrete tools (where possible) as modular kernel extension and/or refinement, recognizing each of them as element of a (at least one) BPM product line.

In this way, one could succeed really to 'distill the essential features of the many workflow management systems' [46, p. 9] and would provide a scientific framework for an explicit, truly specific-language-and-provider-independent 'unbiased comparison of different approaches to the specification of executable business processes' [46, p. 9]. It would identify families of kernels of 'essential' BPM tool features and clarify which feature extensions each of them needs to provide sufficient expressivity to appropriately model the targeted complex BPM applications.

**Acknowledgments** I wish to thank over two dozen colleagues for encouragement, careful reading and critically commenting upon previous versions of this investigation. I thank them anonymously, not to expose them and to assume full responsibility for the criticism expressed here.

[37] In [18, Sect.2] this is expressed as follows: 'end-to-end control is what business stakeholders need to build process-managed enterprise' resp. 'propagating the information from a value chain perspective to a software-development perspective in a coherent and consistent way'.

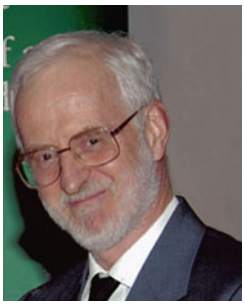
## References

1. Aho, A.V., Lam, M.S., Sethi, R., Ullman J.D.: Compilers: Principles, Techniques, and Tools, 2nd edn. Pearson Education, Inc, India (2006)
2. Ait-Sadoune, I., Ait-Ameur, Y.: Stepwise design of BPEL web services compositions. An event B refinement based approach. In: Software Engineering Research, Management and Applications (SERA 2010). Studies in Computational Intelligence, vol. 296, pp. 51–68. Springer, Berlin (2010)
3. Barros, A., Börger, E.: A compositional framework for service interaction patterns and communication flows. In: Lau, K.-K., Banach, R. (eds.) Formal Methods and Software Engineering. Proceedings of the 7th International Conference on Formal Engineering Methods (ICFEM 2005). LNCS, vol. 3785, pp. 5–35. Springer, Berlin (2005)
4. Batory, D., Börger, E.: Modularizing theorems for software product lines: the Jbook case study. J. Univers. Comput. Sci. **14**(12), 2059–2082 (2008)
5. Börger, E.: Construction and analysis of ground models and their refinements as a foundation for validating computer based systems. Formal Aspects Comput. **19**, 225–241 (2007)
6. Börger, E., et al.: Modeling workflow patterns from first principles. In: Parent, C. (ed.) Conceptual Modeling—ER 2007. Lecture Notes in Computer Science, vol. 4801, pp. 1–20. Springer, Berlin (2007)
7. Börger, E., Sörensen, O.: BPMN core modeling concepts: inheritance-based execution semantics. In: Embley, D., Thalheim, B. (eds.) Handbook of Conceptual Modelling, pp. 287–332. Springer, Berlin (2010)
8. Börger, E., Stärk, R.F.: Abstract State Machines. A Method for High-Level System Design and Analysis. Springer, Berlin (2003)
9. Börger, E., Thalheim, B.: A method for verifiable and validatable business process modeling. In: Advances in Software Engineering. LNCS, vol. 5316, pp. 59–115. Springer, Berlin (2008)
10. Business Process Model and Notation (BPMN): http://www.omg.org/spec/BPMN/2.0, 2011. formal/2011-01-03
11. Christiansen, D., Carbone, M., Hildebrandt, T.: Formal semantics and implementation of BPMN 2.0 inclusive gateways. Pre-Proc. of Web Services and Formal Methods (WS-FM'10) (2010). http://www.itu.dk/people/maca/papers/CD10.pdf
12. Delaware, B., Cook, W., Batory, D.: Product lines of theorems. In: Proc.OOPSLA 2011, Portland, October 2011
13. Denning, P.J., Martell, C.: Great principles of computing. http://cs.gmu.edu/cne/pjd/GP/GP-site/welcome.html (consulted July 26, 2011) (2007)

14. Dijkman, R.M., Dumas, M., Ouyang, C.: Formal semantics and analysis of BPMN process models using Petri nets. Technical Report 7115, Queensland University of Technology, Brisbane (2007)
15. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Inf. Softw. Technol. **50**(12), 1281–1294 (2008)
16. Dumas, M., Grosskopf, A., Hettel, T., Wynn, M.: Semantics of BPMN process models with or-joins. Meersman, R., Tari Z. (eds) et al. OTM 2007 Part I. Lecture Notes in Computer Science, vol. 4803. pp. 41–58. Springer, Berlin
17. Fleischmann, A., Schmidt, W., Stary, C., Obermeier, S., Börger, E.: Subjektorientiertes Prozessmanagement. Hanser-Verlag, Munich (2011)
18. Fleischmann, A., Stary, C.: Whom to talk to? A stakeholder perspective on business process development. Univers. Access Inf. Soc. 1–26 (2011). doi:10.1007/s10209-011-0236-x
19. Frappier, M., Habrias, H. (eds.): Software Specification Methods: An Overview Using a Case Study. HERMES Sci. Publ., Paris (2006)
20. Gao, Y.: BPMN-BPEL transformation and round trip engineering. Technical report, eClarus Software (2006)
21. Graef, N., Tölle, N.: Evaluation, mapping und quantitative reduktion von workflow pattern (control-flow). Bachelor Thesis at Karlsruhe Institute of Technology (AIFB) (2009)
22. Grosskopf, A.: xBPMN. Formal control flow specification of a BPMN based process execution language, pp 1–142. Master's thesis, HPI at Universität Potsdam (2007)
23. Gruhn, V., Laue, R.: What business process modelers can learn from programmers. Sci. Comput. Program. **65**, 4–13 (2007)
24. Hanisch, H.M., Lüder, A.: A signal extension for Petri nets and its use in controller design. Fundamenta Informaticae **41**(4), 415–431 (2000)
25. Kim, C., Batory, D., Khurshid, S.: Reducing combinatorics in testing product lines. In: Proceedings of the Aspect Oriented Software Development Conference. ACM, New York (2011)
26. Kühnle, K., Mayr, E.W.: Exponential space computation of Gröbner bases. In: Lakshman, Y. (ed.) Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation (ISAAC'96), pp 63–71. ACM Press, New York (1996)
27. Lohmann, N.: A feature-complete Petri net semantics for WS-BPEL 2.0 and its compiler BPEL2oWFN, 2007 (August 30)
28. Mayr, E.W., Ritscher, S.: Space-efficient Gröbner basis computation without degree bounds. In: Proceedings of the 2011 International Symposium on Symbolic and Algebraic Computation, pp. 257–264. ACM, New York (2011)
29. Mayr, R.: Process rewrite systems. Inf. Comput. **156**(1–2), 264–286 (2000)
30. McCoy, D.W.: Subject-oriented BPM (S-BPM). Gartner Research Hype Cycle for Business Process Management, 25 July 2011. ID Number: G00214214
31. Metasonic: Metasonic-suite. http://www.metasonic.de/metasonic-suite
32. Mulyar, N., van der Aalst, W., ter Hofstede, A.H., Russell, N.: Towards a WPSL: a critical analysis of the 20 classical workflow control-flow patterns. Technical Report BPM-06-18, BPM Center, http://BPMcenter.org (2006)
33. Ouyang, C., Dumas, M., ter Hofstede, A., van der Aalst, W.: From BPMN process models to BPEL web services. In: ICWS, Los Alamitos, 2006, pp. 285–292. IEEE, New York (2006)
34. Ouyang, C., Dumas, M., van der Aalst, W., ter Hofstede, A.: From business process models to process-oriented software systems: the BPMN to BPEL way. BPM-06-27 at http://BPMcenter.org (2006)
35. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering. Foundations, Principles, and Techniques. Springer, Berlin (2005)
36. Recker, J., Indulska, M., Rosemann, M., Green, P.: Do process modeling techniques get better? A comparative ontological analysis of BPMN. In: Proceedings of the 16th Australasian Conference on Information Systems, Sydney (2005)
37. Recker, J., Mendling, J.: On the translation between BPMN and BPEL: conceptual mismatch between process modeling languages. In: Proc.11th EMMSAD, June 2006
38. Recker J., Mendling, J.: Lost in business process model translations: how a structured approach helps to identify conceptual mismatch. In: Siau, K. (ed.) Research Issues in Systems Analysis and Design, Databases and Software Development, pp. 227–259. IGI Publishing, Hershey (2007)
39. Reisig, W.: Elements of Distributed Algorithms. Modeling and Analysis with Petri Nets. Springer, Berlin (1998)
40. Russel, N., ter Hofstede, A., van der Aalst, W., Mulyar, N.: Workflow control-flow patterns: a revised view. BPM-06-22 at http://bpmcenter.org/reports, July 2006
41. Semiconductor Industry Assoc.: International technology roadmap for semiconductors. Design. http://www.itrs.net/Links/2005ITRS/Design2005.pdf (2005)
42. Signavio: Signavio BPM Academic Initiative. http://www.signavio.com/academic
43. Silver, B.: BPMN method and style: a levels-based methodology for BPM process modeling and improvement using BPMN 2.0. (2009)
44. Störrle, H., Hausman, J.H.: Towards a formal semantics of UML 2.0 activities. In: Proceedings of the Software Engineering 2005 (2005)
45. Strosnider, J.K., Nandi, P., Kumaran, S., Gosh, S., Arsanjani, A.: Model-driven synthesis of SOA solutions. IBM Syst. J. **41**(5), 415–432 (2008)
46. ter Hofstede, A., van der Aalst, W., Adams, M., Russell, N. (eds.): Modern Business Process Automation. Springer, Berlin (2010)
47. Uzuncaova, E., Khurshid, S., Batory, D.: Incremental test generation for software product lines. IEEE Trans. Softw. Eng. **36**(3), 309–322 (2011)
48. van der Aalst, W., Hirnschhall, A., Verbeek, H.: An alternative way to analyze workflow graphs. In: Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE02). LNCS, vol. 2348, pp. 535–552. Springer, Berlin (2002)
49. van der Aalst, W., ter Hofstede, A.: YAWL: yet another workflow language. Inf. Syst. **30**(4), 245–275 (2005)
50. van der Aalst, W., ter Hofstede, A.: Workflow patterns home page. http://www.workflowpatterns.com, created and maintained since 1999
51. van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow patterns. Distrib. Parallel Databases **14**(3), 5–51 (2003)
52. Voelzer, H.: A new semantics for the inclusive converging gateway in safe processes. Technical Report RZ 3791, IBM Research Zürich (2010)
53. Wei, W.: A translation from BPMN to Event-B. Manuscript (2010)
54. Weidlich, M., Decker, G., Grosskopf, A., Weske, M.: BPEL to BPMN: the myth of a straight-forward mapping. In: On the Move to Meaningful Internet Systems: OTM 2008, Part I. LNCS, vol. 5331, pp. 265–282. Springer, Berlin (2008)
55. Weissbach, M., Zimmermann, W.: Termination analysis of business process workflows. In: Proceedings of the 5th International Workshop on Enhanced Web Service Technologies, pp. 18–25. ACM, New York (2010)
56. Weske, M.: Business Process Management. Springer, Berlin (2007)
57. Wimmel, H., Priese, L.: Petri-Netze, 2nd edn. Springer, Berlin (2008)

58. Wohed, P., van der Aalst, W., Dumas, M., ter Hofstede, A., Russel, N.: On the suitability of BPMN for business process modelling. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) Business Process Management. 4th International Conference BPM 2006. LNCS, vol. 4102, pp. 161–176. Springer, Berlin (2006)
59. Wu, Y., Hernandez, F., Clarke, P.J., France, R.: A DSML for coordinating user-centric communication services. In: 34th Annual IEEE Computer Software and Applications Conference (COMPSAC 2011), Munich, 18–21 July 2011. IEEE, New York (2011)
60. zur Muehlen, M., Recker, J.: How much BPMN do you need? Posted at http://www.bpm-research.com/2008/03/03/how-much-bpmn-do-you-need/
61. zur Muehlen M., Recker, J.: How much language is enough? Theoretical and practical use of the Business Process Modeling Notation. In: Bellahsène, Z., Léonard, M. (eds.) 20th International Conference on Advanced Information Systems Engineering (CAiSE 2008). LNCS, vol. 5074, Montpellier, June 2008, pp. 465–479. Springer, Berlin (2008)

## Author Biography



**E. Börger** Studies of philosophy, logic and mathematics in Paris (Sorbonne), Louvain (Belgium) and Münster (Germany) 1965–1971. Doctoral degree (1971), Habilitation (1976), Dozent in mathematics 1976–1978 at U of Münster. Professor in computer science at U of Salerno (Italy) 1972–1976, Dortmund (Germany) 1978–1985, Udine (Italy) 1982/1983 and since 1985 Pisa (Italy). Sabbaticals spent at IBM, Siemens, Microsoft, SAP, ETH Zürich. (Co)Author of five books and of over 100 research papers in logic and computer science. Current research interest in rigorous methods and their industrial application for the design and the analysis of hardware/software systems. Humboldt Research Award 2007, *Festschrift* LNCS 5115, member of Academia Europaea.