Egon Börger (Pisa)

Business Process Modeling: Standards or Accurately Modeled Tools?

A Case Study: BPMN, Workflow Patterns, YAWL

boerger@di.unipi.it
Università di Pisa, Dipartimento di Informatica, Italy

The type of problem: semantics in computing

Provide a precise description of the behavior ('meaning') of objects of computing (read: algorithms, programs or more generally systems involving computation) which users can

- understand
- rely upon

in the sense that the objects (read: implementations) behave in accordance with the description

Outstanding examples of such objects are

- models of computation
- programming languages
- database management systems
- business process modeling and management systems

Characteristics of approaches to the problem

Separation of human-oriented high-level definitions from implementations (typically mediated by compiling), e.g.

- in theory of programming: mathematical (abstract operational, denotational, axiomatic) definition of the meaning of programs vs their execution behavior ('meaning' of compiled code and eventually of runs on physical machines)
- in database theory: mathematical definition of data and data operations vs their (eventually physical) implementation, using e.g.
  - logic, e.g. relational algebra, first-order logic, logic programming following Codd's interpretation of data as sets of tuples
  - richer data structures, e.g. models with complex values (e.g. defined by integrity constraints) or semantic data models (including handling of meta-data) or oo data models (including behavioral aspects)

Common concern: importance of human-oriented models which define program/data operations in correspondence to their implementations

Semantics of business processes and BPM systems

Instance of the general problem (view 'process' as 'program'):

- how to reliably relate the user understanding of a business process to the machine execution of the process
  - BP modeling and lifecycle concepts well-known from sw engineering
    - requirements ('ground') models and their (possibly automated) refinement to executable code for both system design and analysis
      - verification and validation of models (including code) at each level of abstraction (also at runtime)
- additional challenges:
  - huge diversity of application domains
  - typically a business process involves actions of humans
  - most users have no familiarity with basic notions of computing

Early focus on control flow

Initially considered essentially as a control flow (execution order, also called *sequence flow*) problem

- abstracting widely from underlying data and communication flow and from used resources, using for BP modeling mostly
  - original version of EPCs (Event Driven Process Chains)
    - in German EPK (Ereignisgesteuerte Prozesskette)

    with some later (also not precisely defined) additions to annotate process data and participants
  - workflow languages
    - e.g. Workflow Management Coalition (informal) standard
      - see www.wmfc.org/
  - Petri nets
    - special case: workflow nets where each task is on a path from unique start to unique end place

Focus on control flow (Cont'd)

Later higher-level approaches to workflow management, e.g.

- Activity Diagrams in OMG standard for UML 2.0
- BPMN 2.0 (2011)
- workflow patterns proposed by Workflow Pattern Initiative (since 1999)

target more general (machine supported or automated) BP management concepts but still share insufficient support for data, communication, resource aspects (in particular when it comes to include humans as actors)

NB. BPEL (since 2003) has a detailed communication mechanism to describe interaction between processes but is considered as low-level (e.g. BPMN target) implementation language.

How to describe BPMs in an appropriate way?

From: Ch.1 of A.H.M.ter Hofstede, W.M.P.van der Aalst, M.Adams, N.Russell (Eds): Modern Business Process Automation. Springer 2010

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

- 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.
  - ... not concerned with *expressive power* but with *suitability*, a notion that refers to the alignment between a modeling language and a problem domain.
- Standardization efforts in the field have essentially failed. (p.5)

The surveyed examples include:

- XPDL (Workflow Management Coalition, 90'ies, www.wmfc.org/), BPEL, BPMN, EPC, UML Activity Diagrams

Standardization failure: some BPMN 2.0 deficiencies (1)

- numerous *ambiguities, underspecifications, overspecifications*, e.g.
  - underspecified *lifecycle* concept, in particular in relation to underspecified *interruption* mechanisms:
    - exception handling
      - stack-like nesting?
      - in case of multiple enabled nested interrupt (e.g. boundary timer) events: should one or all of them (in which order?) fire?
      - no predefined interpreter exception types
    - cancelation (e.g. scope for subprocesses or running instances of multiple instances within a process instance)
    - compensation for transactions
  - incomplete activity *completion* criteria (e.g. for simultaneous completion of multiple subprocess instances)

Standardization failure: some BPMN 2.0 deficiencies (2)

- underspecified *expression evaluation*
  - when to evaluate (e.g. event) expressions
    - dependent on event type it could (probably should) be either before/at process start or upon state change or when token becomes available
  - in case multiple events use a same non-deterministic expression definition:
    - is the evaluation the same for each event?
    - are all the events which listen to the common definition triggered simultaneously or only one (which consumes the event)?
  - exclusive gateway expressions 'to be evaluated in order'
    - in which order? how displayed?
- undefined interaction of *transient and persistent triggers* (e.g. when do transient triggers become obsolete?)

Standardization failure: some BPMN 2.0 deficiencies (3)

- poor *data* support (no general notion of state) makes data management of an executable version compiler (data model) dependent
  - data objects (associated to activities/sequence flow) used only informally, in particular with underspecified assumptions on input/output selection at task nodes or sequence flows
  - variables (called properties), like most other attributes, are graphically invisible though they influence the diagram behavior
- poor support for *resource* management
  - via lanes (actors, roles, ...) or performers of user/manual tasks
- poor support for system *structure* (risk of 'spaghetti diagrams')
- underspecified *communication* mechanism
- underspecified *concurrency and interaction* model (e.g. role of independent (not embedded) subprocesses)
- OR-join difficult to understand and use
- unclear (multiple) token model and its impact on semantics definition

Standardization failure: some BPMN 2.0 deficiencies (4)

- *plethora of interdefinable constructs*
  - instead of a core of independent BPMN constructs in terms of which other constructs can be defined (see EB/Thalheim LNCS 5316)
  - fuzzy overlapping prevents 'closed' description of single constructs in one place
    - comprehension of various constructs necessitates simultaneous consideration of numerous sections of the standard document
  - statistical analysis (Michael zur Mühlen/Jan Recker 2008) showed:
    ... the average BPMN model uses less than 20% of the available vocabulary ...
    Only five elements (normal flow, task, end event, start event, and pool) were used in more than 50% of the models we analyzed.

  NB. BPMN 2.0 has more than 116 graph symbols.

Standardization failure: some BPMN 2.0 deficiencies (5)

- *unmediated gap between conceptual and executable BPMN models*
  - failure to provide seemless refinement mechanism of conceptual to executable models (to guarantee reliability of implementation)
    - executable BPMN models hard to grasp for human readers
  - lack of standardized view mechanisms to extract abstract (e.g. user or management) models from detailed (e.g. sw) models
    - e.g. communication specific abstractions of WF concepts in WF-CML (Clarke& France et al Compsac11)

NB. Practical relevance of this problem for the use of BPMN evidenced by statistical analysis (Michael zur Mühlen/Jan Recker 2008):

... we 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).

Standardization failure: full BPMN 2.0 implementable?

BPMN claims to (it should!) provide models which serve three purposes:

- *analysis* for high-level management support (conceptual model)
- *implementation* as spec of software requirements (executable model)
- *use* for process management and monitoring (user model)

They are claimed to (indeed should!) be seemlessly linkable via refinements or transformations to executable (e.g. BPEL) code.

But underspecification/ambiguities imply compile time *behavior-relevant decisions* for the executable model which remain *invisible in the conceptual/user model*.

Standardization failure: full BPMN 2.0 not implementable!

As a consequenceof underspecification/ambiguities:

BPMN 2.0 fails to be implementable with the claimed 'generality'

- without limiting numerous 'variation points' by imposing semantically significant restrictions of concepts which are left underspecified in the standard

As a consequence 'standard conform' models

- usually cover only a subset of BPMN
- *hinder reliable communication* between stakeholders
- are *not guaranteed to be interoperable* and portable (platform independent)
- *limit verification efforts*: verification under which assumptions?

Gretchenfrage: a 'reference implementation' becomes the standard?

Workflow Patterns as measure to evaluate BPM approaches?

The two senior book authors at Eindhoven/Queensland UT started in 1999 the *Workflow Patterns Initiative* (www.workflowpatterns.com) to

- describe process modeling requirements in an *implementation independent* manner (op.cit.p.5)
- to distill the *essential* features of the many workflow management systems that existed. This would allow an *unbiased* comparison of different approaches to the spec of executable BPs & provide the basis for the adaptation & refinement of exisiting techniques as well as supporting the development of new approaches. (op.cit.p.9)

But ...

- How are WPs justified?
- How are WPs described? Suitably to 'describe proc modeling requs'?
- WP-based comparison unbiased? WPs a 'suitable' eval measure?
- Is there really consensus to use WPs to evaluate BPM systems?

Workflow/Data/Resource Patterns: Foundational Problems

- no rational foundation of choice/classification but an *exponential growth* (ending where?) produced by a small group:
  - 20 in 2003: Wil M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, A.P. Barros: Workflow Patterns. Distributed and Parallel Databases 13 (4) (2003) 5-51
  - 46 in 2006: N. Russel, A.H.M. ter Hofstede, Wil M.P. van der Aalst, N. Mulyar: Workflow Control-Flow Patterns. A Revised View. BPM Center Report BPM-06-22
  - 126 in 2010: A.H.M.ter Hofstede, W.M.P.van der Alst, M.Adams, N.Russell (Eds): Modern Business Process Automation (p.25)
- no statistical underpinning: how often WPs used in real-life BPs?
- natural language descriptions in many places ambiguous or incomplete or overspecified by implementation features (mostly belonging to coloured Petri nets)
  - NB. formalization in (ad hoc) WPSL (BPM-06-18) not really helpful

WPI biased to quantity (neglecting conceptual simplicity)

- 20/46/126 WPs not fundamental (more molecule than element like)
  - do not 'distill the essential features of the many workflow management systems'

  but are all definable from a small set of more basic patterns
  - all 46 WPs from 2006 rigorously modeled by parameter *instantiation of 8 natural generic schemes*
    - most of which represent well known—for the targeted WPs appropriately parameterized— sequential or concurrent programming constructs (EB in LNCS 4801, 2007)
  - Graef/Tölle (KIT 2009) modeled 43 WPs from 2006 in S-BPM
    - mapping from 25 more basic (appropriately generalized) 'choreography patterns' (reduction by 42%, further improvement considered to be possible)
  - similar reduction to be expected for today's 126 (including data, resource, exception) patterns

WPs claimed as benchmark to evaluate BPM systems

WPI www.workflowpatterns.com/vendors corner declares (May 2011)

> that the evaluation of workflow products on this site only considers direct support, 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*. In other cases, it is impossible to work around the problem and the designer has to resort to developing additional software.

tacitly assuming that missing 'direct support' is 'a problem'.

Is this assumption reasonable, scientifically or pragmatically?

Are WPs good benchmark to evaluate BPM systems?

- Restriction to 'direct support' misjudges definability as something negative.
  - Definability is a royal road to modularity and complexity reduction and by no means an expedient to 'work around the problem'.
  - The authors themselves present having in the 'reference implementation' (YAWL)

    a minimal set of constructs, rather than a construct-per-pattern approach (p.14)

    as an advantage (see below): thus (some) WPs are not directly supported but have a definition in YAWL?!
- How often is support of which WPs needed in real-life applications?
- WPI vendors corner lists 13 evaluated tools (out of > 160):
  - How representative are these tools?
  - Any guarantee for unbiased evaluation, free of 'vested interest' of (e.g. financial contribution to) the group behind WPI?

Proposed remedy: an 'academic' reference implementation

YAWL ... a reference implementation that supports the WPs (p.11)

developed without the pressures of vested interest and a sole focus on providing *comprehensive support for the Workflow Patterns* (p.14)

However:

- M. Weske noted some badly WP-benchmarked YAWL pattern impls!
- Support for WPs appears as a vested interest of the group behind the BPM Center & its WPI which markets (a growing set of) WPs.
- Vicious circle? a) YAWL justified as support for WPs but b) WPs
  - which have no 'natural' (formalism independent) semantical foundation in their own (see above)

  are semantically defined by YAWL (a reference implementation!)

Qu: compared to other academic/open-source workflow/BPM systems:

Is YAWL appropriate for BPM?

Is the semantical foundation of YAWL appropriate for BPM?

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. (op.cit.p.10)

...the semantics has been defined in terms of a large Colored Petri net (p.15)

Why coloured Petri nets? The authors' reasons (p.10):

three reasons why Petri nets would make a good candidate...for workflow specification...

- the *graphical nature* of Petri nets,
- their *explicit representation of the notion of state*, and
- the existence of *analysis techniques*

Are these good reasons?

Petri/Workflow/Reset net states/steps sufficient for BPs?

- not general enough notion of state to deal with BPs
  - encodings are needed for representation of complex BP objects
  - reset nets provide a token-based encoding of the abstract concept (called 'cancellation') of process disactivation

    $\textbf{forall } p \in \mathit{ToBeStopped}(\mathit{params})\ \ \textsc{Disactivate}(p)$

  contradicting the authors' request for

  suitability, a notion that refers to the alignment between a modeling language and a problem domain (op.cit.p.9)

Petri/Workflow/Reset net states/steps sufficient for BPs? (2)

- insufficient computational power (transition step): difficulties to describe *non-local* or *truly concurrent* or *recursive* behavior, e.g.
  - OR-join, cancelation (triggering reset YAWL construct)
  - multiple concurrent task instances, synchronization of active branches
    - see also problems with mapping BPMN diagrams and UML 2.0 Activity Diagrams to Petri nets (references below)
  - nondeterministic execution model hides inappropriate implicit 1-core interleaving assumption ('no 2 transitions fire simultaneously')
  - general recursion (a priori unbounded loops) not covered
    - Petri $= (P; P) \not\subseteq (P, S; P, S)$ Process Rewrite Syst (Mayr 2000)

  Consequence: numerous workflow net additions needed for BP features which are not covered by mere token-based Petri net model, e.g.
  - YAWL *task start* classification: automatic, user, external, time
  - open WF net extension (with strong fairness assumption) to include *communication* via messages

Petri/Workflow/Reset net states/steps sufficient for BPs? (3)

- unnecessary complexity of definitions of semantics for various high-level BP concepts in YAWL, due to the lack of data types which are appropriate at the BPM intersection of business and information technology, e.g. for
  - multiple instance tasks with upper/lower runnable instance bounds, threshold for completion, dynamic/static start attribute
  - various resource allocation strategies

Conclusion: Petri/Workflow/Reset nets appear as illustrative examples of languages which (in the authors' wording)

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

'Existing Petri net analysis techniques' suitable for BPs?

- How 'suitable' are the advocated existing tool supported Petri net based analysis techniques for real-life (not academic examples of) BPs?
  - Focus on traditional Petri net properties—sequence flow (abstraction from data, communication, resources), fairness, liveness—appropriate for high-level BPM analysis?
    - e.g. liveness analysis irrelevant in presence of timeouts and fairness (if an issue) usually delegated to the scheduler
    - *Support needed for analysis of BP-process tailored abstractions*!
  - How many of the advocated tool supported Petri net based analysis techniques have been ported to the needed YAWL extension?
    - e.g. Petri net tools usually assume (for fairness reasons) that loops terminate: this excludes possible execution paths of relevant real-life BP behavior (Weissbach/Zimmermann 2010) which therefore is only incompletely covered by such analysis tools

'Graphical nature of Petri nets' relevant for BPs?

- Graphical nature of Petri nets
  - contributes only in a minor way to graphical elements needed for satisfactory BPM language (see numerous YAWL constructs)
  - often complicates process representation unnecessarily by low-level (token-focussed workflow control) details
    - e.g. compare relatively large Petri nets for (simple!) network algorithms in Reisig's Distributed Algorithms book (1998) with their small models in AsmBook Ch.6.1

3 criteria advocated to 'uniquely position' YAWL not satisfied

'YAWL has a number of features that position it uniquely in the crowded field of BPM' (op.cit.p.14-16)

- '... a minimal set of constructs, rather than a construct-per-pattern approach'
  - but minimality is claimed not proved (and should hold directly for WPs, why for the 'reference implementation'?)
    - S-BPM has fewer/more fundamental constructs with WP independent epistemologically well-founded rational
  - consistent with request for 'direct support' for patterns?
- 'sophisticated flexibility support ... *exceptions* may arise ... processes may evolve over time due to *changes* in a business and/or its environment'
  - but flexibility is supported also by other approaches
    - e.g. S-BPM has 2 simple general extension constructs covering exceptions and modularity (including evolutionary changes)

3 criteria advocated to 'uniquely position' YAWL not satisfied

- 'a formal foundation... 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,'
    - NB: this connection remains unclear in op.cit. where—after 676 pages of explanations of YAWL—the 'large Colored Petri net' claimed to define the semantics of YAWL does not appear
      - Qu: how can such a Petri net exist since YAWL contains 'patterns that Petri nets have difficulty expressing'?
  - 'or (2) the formalization is not generally accepted, and certainly not by a standard body'
    - *Are Colored Petri nets* generally accepted? (or by a BPM standard body?), i.e. outside the Petri community and comprising the BPM stakeholders?

Side remark: questioning 'uniquely positioning criteria'

There are other (also academic and open source) BPM systems with features of interest to the practitioner that are not mentioned in the list of those claimed to 'position YAWL uniquely'.

An example:

- Tool offered by Signavio
  - with free use for teaching and research offered through the BPM Academic Initiative signavio.com/academic

  features:
  - modeling in different languages
  - integration of different workflow engines offered by other providers
  - integration of various analysis tools
  - open source engine with native support for a subset of BPMN 2.0 (lightweight Java approach)

Essential criteria for evaluating BPM tools

- support, for BP descriptions, of their
  - correct understanding (during development and use)
  - faithful implementation via systematic refinements
  - effective management (evaluation, monitoring, change)
- support for
  - abstraction
    - for the development of models out of concrete real-life problem situations
  - modularization with precise interfaces and behavior models
    - to achieve structured, easy-to-change system descriptions
- precise foundation a practitioner can understand and rely upon

Why not accurate modeling and development of industrial tools
which satisfy the above criteria?

Industrial tools with simple precise semantics?

Apparently all current (Gartner Group: ca. 160) industrial BPM tools come without reliable (precise and easy to understand) model of the underlying semantics

- one exception: PASS tool of S-BPM distinguishes itself by a simple epistemological and mathematically stable foundation for:
  - clear separation of mere *control flow* from actions
  - explicit distinction of *internal actions* from communication
  - uniform treatment of both *a/synchronous communication*
  - clear interface to underlying *data structures* of executing agents
  - natural *modular composition* constructs (including stepwise extensions of normal or interrupt behavior)

  A rigorous simple mathematical model defines the semantics of these features directly, in a modular fashion, using stepwise refinement

See A. Fleischmann, W. Schmidt, C. Stary, S. Obermeier, E. Börger: *Subjektorientiertes Prozessmanagement* Hanser-Verlag München, 2011

Challenge for rigorous foundation of BPM systems

- identify a common kernel of major BPM tools in terms of control, actions, communication
- build a precise behavioral model for this kernel which reflects the kernel concepts directly (avoiding encoding into any form of a priori determined description language other than the general language of computing)
- model concrete tools as modular kernel extension and/or refinement
  - thus providing a scientific framework for explicit, truly specific-language-and-provider-independent comparison and evaluation (including abstract performance analysis) of different tools

Reference

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

- J. Software and Systems Modeling 2011
  DOI 10.1007/s10270-011-0214-z

A few more references (standing for many)

On problems with the BPMN standard:

- J. Recker, M. Indulska, M. Rosemann, P. Green: Do process modeling techniques get better? A comparative ontological analysis of BPMN. Proc. 16th Australasian Conf. on Information Systems, Sydney 2005
- P. Wohed, W.M.P.van der Aalst, M. Dumas, A.H.M. ter Hofstede and N. Russell: On the suitability of BPMN for business process modelling. LNCS 4102 (2006) 161-176 (Proc.4th Int.BPM Conf.)
- E. Börger and O. Sörensen: BPMN Core Modeling Concepts: Inheritance-Based Execution Semantics. In: Handbook of conceptual modelling, Springer 2010
- E. Börger and B. Thalheim: A Method for Verifiable and Validatable Business Process Modeling. LNCS 5316 (2008)

A few more references

On Problems to map BPMN/UML 2.0 ActDgms to Petri nets:

- R. M. Dijkman and M. Dumas and C. Ouyang: Formal Semantics and Analysis of BPMN Process Models using Petri Nets. QUT TR 7115 (2007). See also 'Semantics and analysis of business process models in BPMN' by the same authors in: Information and Software Technology, 50(12) 1281- 1294, 2008.
- Alexander Grosskopf: Formal Control Flow Specification of a BPMN based Process Execution Language. Universität Potsdam, HPI, July 2007, p. 1-142

A few more references

On Problems to map BPMN/UML 2.0 ActDgms to Petri nets (2)

- David Christiansen, Marco Carbone and Thomas Hildebrandt: 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
- Harald Störrle and Jan Hendrik Hausman: Towards a Formal Semantics of UML 2.0 Activities. Proc. Software Engineering 2005
- M.Weissbach and W. Zimmermann: Termination analysis of business process workflows. Proc. 5th Int. ACM Workshop on Enhanced Web Service Technologies (2010) 18-25
- R. Mayr: Process rewrite systems. Information and Computation 156 (1-2) (2000) 264-286

A few more references

On reduction of BPMN or Workflow Pattern constructs:

- E. Börger: Modeling Workflow Patterns from First Principles. Springer LNCS 4801 (2007) 1-20
- Norbert Graef and Nils Tölle: Evaluation, Mapping und quantitative Reduktion von Workflow Pattern (Control-Flow).
  Bachelor Thesis KIT/AIFB 2009 (43 WPs represented in S-BPM. Suggests reduction to (less than) 18 WPs).
- Michael zur Muehlen and Jan Recker: On use of BPMN constructs: How much BPMN do you need?
  http://www.bpm-research.com/2008/03/03/how-much-bpmn-do-you-need/
  See by the same authors: How Much Language Is Enough? Theoretical and Practical Use of the Business Process Modeling Notation. LNCS 5074, pp. 465479, 2008

Acknowledgements

Stephan Borgert, Hagen Buchwald, Matthes Elstermann, Albert Fleischmann, Stephan Mennicke, Detlef Seese, Ove Sörensen, Bernhard Thalheim, Mathias Weske, Wolf Zimmermann

