A Note on Analysis of BPMN Diagrams

Antoni Ligęza

AGH University of Science and Technology, Department of Automatics

Abstract: BPMN has recently become a *de facto* standard for modeling and design of complex software intensive processes. It is widely used not only in the Business Process domain. Numerous tools supporting visual edition have been developed. Despite its unquestionable advantages the semantic analysis of logical properties seems to be one of the weaknesses of this formalism. In order to assure reliable process execution the overall structure of the graph and its logical operation should be verified.

Keywords: Business Process Modeling Notation, BPMN, Business Rules, Rule-Based Systems, System Analysis, System Verification, Safety, Reliability, Efficiency, Correctness

1. Introduction

Modern decision support and process management systems become more and more complex. In numerous areas of business and technological applications incredibly sophisticated software-intensive systems can be observed. Such systems are often made intelligent by incorporation of *Artificial Intelligence* tools and techniques [1]. In areas such as process management, control, optimization, diagnosis and management use of intelligent systems may be crucial for achieving technological superiority [2]. Application areas range from human-computer interaction systems (e.g. urban traffic control), through production management, to computer-intensive applications (such as Internet, telecommunication, navigation).

In order to assure smooth performance of such systems, some qualitative requirements are imposed on them from an early design stage. Most typical approach consists in specifying a demand for **safety** – whatever happens, the system should assure that nothing harmful will happen, **reliability** — the system should do its job right, and **efficiency** — the system should be as productive as possible.

In order to make easier design and development system of high quality advanced software engineering approaches are in use. One direction consists in employing *visual design tools* [3–5]. This trend is observed especially within the domains of Rule-Based Systems including Business Rules [6, 7]; HaDEs is an example of a modern dedicated environment for visual design [8] and HQEd is a visual rule editor [4]. Also application of UML can be considered [9]. Another possibility consists in structuring the rule-base into easy to analysis and inference control components [10]. Both methods can be applied for analysis and control of Business Rules and Business Processes [3].

Another direction consists in application of formal methods and logical verification of required properties [11–14]. In this paper an extension of such methods coming from the area of verification of Rule-Based Systems towards BPMN is considered.

2. BPMN Systems Quality

BPMN stands for *Business Process Modeling Notation*. It is composed of a set of graphical symbols, such as links modeling workflow, splits and joins, events and boxes symbolizing data processing. It constitutes a transparent visual tool for modeling complex processes promoted by OMG [15]. In Fig. 1 an example BPMN diagram for a decision process is specified.

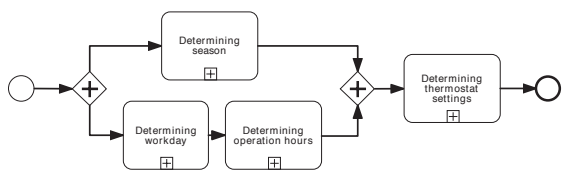

Fig. 1. An example BPMN diagram — specification of a decision support system
Rys. 1. Przykładowy diagram BPMN — specyfikacja systemu wspomagania decyzji

Since BPMN allows for modeling data and control flow (including activities, splits and joins of data flow, conditional operations, loops, event-triggered actions, paths and communication processes) [16], in practice arbitrarily complex systems can be modeled. On the other hand, however, attempts at more rigorous analysis become problematic. A first attempt can be done with use of Business Rules [17]. However, it is in general hard to assure correct behavior of the system through simple rule-based off-line analysis since the workflow specification is more complex and the semantics is not defined in terms of logic. Translation to verifiable languages can be another possibility [14], but due to numerous components in BPMN 2.0 it becomes tedious and problematic.

A BPMN diagram can model arbitrarily complex processes. It is crucial to assure correctness of a BPMN specification. Apart from *external consistency* validation (i.e. whether or not the diagram models correctly the external system in a *complete* and *consistent* way, does not introduce any non-existent features, and there is *isomorphism* between those two), a crucial issue is the *internal*

consistency requirement. It is further composed of the following requirements: (i) **structural correctness**, (ii) **logical correctness**, and (iii) **reachability** of each end point.

The *structural correctness* is defined as a set of requirements for the so-called *well-formed BPMN diagram* [18]. In fact, structural correctness allows to define the syntax of meaningful BPMN diagrams. Example constraints are as follows: *a starting point has no incoming links, an end point has no outgoing links, any task node has exactly one incoming and one outcoming link, etc.* More details can be found in [18]. The diagram presented in Fig. 1 is structurally correct.

However, even having correct structure, the process can gone wrong due to unserved data or wrong workflow control, for example. For example, if the output produced by one task is not accepted by the next task in turn, the process cannot be executed. Similarly, if at some split node the current data does not satisfy any split condition, the control will not be passed further on. An attempt of some minimal requirements for logical correctness must include (i) correct work of process components (tasks), (ii) assuring data flow between tasks, (iii) correct work of splits, (iv) correct work of merge nodes, and finally — (v) termination of the overall process; for details see [18].

Finally, reachability refers to dynamic properties of the system; each termination point must be reachable (at least potentially); in other case it is useless, and the goal assigned to it will never be reached. For intuition, reachability means that each part of the BPMN diagram defines a live workflow — one that can be executed for certain data.

Reachability can be modeled with use of tokens [19]. It can be also defined in a recursive way as follows [18]:

- any start node s is trivially reachable; it is simultaneously initiated by assigning it current input data satisfying formula ϕ;
- consider a task T defined with some action/algorithm; its output node is reachable if and only if its input node is reachable with restriction formula ϕ, such that the algorithm always produces the output;
- the case of split nodes:
  - an output node of an exclusive split node corresponding to a branch defined by switching condition q_i is reachable if and only if its input node is reachable with restriction formula ϕ, and $\phi \models q_i$; note that for a given ϕ *only one* output node can be reachable, and for correct exclusive split nodes exactly one can be reached for a particular current data;
  - an output node of an inclusive split node corresponding to a branch defined by switching condition q_i is reachable if and only if its input node is reachable with restriction formula ϕ, and $\phi \models q_i$,
  - any output node of a parallel split node is reachable, if and only if its input node is reachable.
- the case of merge nodes:
  - the output node of an exclusive merge node is reachable if and only if at least one of its input nodes is reachable, and the conjunction of input formula ϕ_i and the link condition p_i is satisfied,
  - the output node of an inclusive merge node is reachable if and only if at least one of its input nodes is reachable, and the conjunction of input formula ϕ_i and the link condition p_i is satisfied,
  - the output node of a parallel merge node is reachable if and only if all of it input nodes are reachable, and the conjunctive input formula is satisfiable.

Some further requirements may refer to *minimal representation*, *optimal decomposition*, *robustness*, *optimal results* and *optimal execution*. In order to answer these questions one must (i) assure correct work of all the components (activities/processes), (ii) assure correctness of data flow between components, (iii) assure correct inference control (w.r.t. split and join operations), (iv) check if the static structure of the diagram is correct and, finally (v) check if all this combined together will work, i.e. the inference process is not blocked at some point (e.g. due to a deadlock).

3. Concluding Remarks

A note on requirements for correct specification and development of BPMN diagrams is presented. Both structural, logical, and dynamic characteristics are taken into account. The main focus is on specification of correct components and correct dataflow as well as correct flow control. Global termination conditions are specified in a recursive way.

Acknowledgment

Research supported from the Polish Ministry of Science and Education under the BIMLOQ Project N N516 422338, 2010-2012.

References

1. Tadeusiewicz R. (2011): *Introduction to intelligent systems*, [in:] Wilamowski B.M., Irwin J.D. (eds.): *Intelligent systems, The Electrical Engineering Handbook Series*, The Industrial Electronics Handbook, 1–1–1–12, Boca Raton, London, New York, CRC Press Taylor & Francis Group, second edition.
2. Tadeusiewicz R. (2005): *Sztuczna inteligencja jako narzędzie budowy przewagi konkurencyjnej*, [in:] Duda J.T. (ed.): *Systemy informatyczne i metody obliczeniowe w zarządzaniu*, Wyd. Naukowo-dydaktyczne AGH, Kraków, 17–26.
3. Kluza K., Nalepa G.J., Łysik Ł. (2010): *Visual Inference Specification Methods for Modularized Rulebases. Overview and Integration Proposal*, [in:] Nalepa G.J., Baumeister J. (eds.): 6th Workshop on Knowledge Engineering and Software Engineering (KESE2009) at the 32nd German conference on Artificial Intelligence, September 21, 2010, Karlsruhe, Germany, 6–17.
4. Kaczor K., Nalepa G.J. (2009): *Extensible design and verification enviroment for XTT rule bases*, [in:] CMS'09: Computer Methods and Systems: 7th conference, 26–27 November 2009, Kraków, Poland, AGH University of Science and Technology, Oprogramowanie Naukowo-Techniczne.
5. Nalepa G.J. (2009): *Languages and Tools for Rule Modeling*, [in:] Adrian Giurca K.T.Dragan Gasevic (ed.): *Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches*, IGI Global, Hershey, New York, 596–624.
6. Ligęza A., Nalepa G.J. (2011): *A study of methodological issues in design and development of rule-based*

systems: proposal of a new approach, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1(2), 117–137.
7. Nalepa G.J., Ligęza A. (2010): *HeKatE Methodology, Hybrid Engineering Of Intelligent Systems*, "International Journal of Applied Mathematics and Computer Science" 20(1), 35–53.
8. Kaczor K., Nalepa G. J. (2009): *HaDEs – Presentation of the HeKatE Design Environment*, [in:] Baumeister J., Nalepa G.J. (eds.): 5th Workshop on Knowledge Engineering and Software Engineering (KESE2009) at the 32nd German conference on Artificial Intelligence, September 15, 2009, Paderborn, Germany, 57–62.
9. Nalepa G.J. (2009): *XTT Rules Design and Implementation with Object-Oriented Methods*, [in:] Lane H.C., Guesgen H.W. (eds.): FLAIRS-22: Proceedings of the twenty-second international Florida Artificial Intelligence Research Society conference, 19–21 May 2009, Sanibel Island, Florida, USA, FLAIRS, AAAI Press, Menlo Park, California, 390–395.
10. Bobek S., Kaczor K., Nalepa G.J. (2010): *Overview of Rule Inference Algorithms for Structured Rule Bases*, Gdansk University of Technology Faculty of ETI Annals 18(8), 57–62.
11. Ligęza A. (2006): *Logical Foundations for Rule-Based Systems*, Springer-Verlag, Berlin, Heidelberg.
12. Ligęza A., Nalepa G. J. (2009): *Rules verification and validation*, [in:] Adrian Giurca K. T.Dragan Gasevic (ed.): *Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches*, IGI Global, Hershey, New York, 273–301.
13. Nalepa G., Bobek S., Ligęza A., Kaczor K. (2011): *HalVA - Rule Analysis Framework for XTT2 Rules*, [in:] Bassiliades N., Governatori G., Paschke A. (eds.): *Rule-Based Reasoning, Programming, and Applications*, "Lecture Notes in Computer Science" volume 6826, Springer Berlin / Heidelberg, 337–344.
14. Szpyrka M., Nalepa G.J., Ligęza A., Kluza K. (2011): *Proposal of Formal Verification of Selected BPMN Models with Alvis Modeling Language*, [in:] Brazier F. M., Nieuwenhuis K., Pavlin G., Warnier M., Badica C. (eds.): "Intelligent Distributed Computing V, Studies in Computational Intelligence", Springer-Verlag [in press].
15. OMG (2011): *Business Process Model and Notation (BPMN): Version 2.0 Specification*, Technical Report formal/2011-01-03, Object Management Group.
16. Ouyang C., Dumas M., ter Hofstede A.H., van der Aalst W.M. (2006): *From BPMN Process Models to BPEL Web Services*, IEEE International Conference on Web Services (ICWS'06).
17. Nalepa G. J., Kluza K., Ernst S. (2011): *Modeling and Analysis of Business Processes with Business Rules*, [in:] Beckmann J. (ed.): *Business Process Modeling: Software Engineering, Analysis and Applications, Business Issues, Competition and Entrepreneurship*, Nova Publishers.
18. Ligęza A. (2011): *BPMN — A Logical Model and Property Analysis*, "Decision Making in Manufacturing and Services" 5(1-2), 5–15.
19. Götz M., Roser S., Lautenbacher F., Bauer B. (2009): *Token Analysis of Graph-Oriented Process Models.*

Przyczynek do analizy diagramów BPMN

Streszczenie: BPMN staje się powoli standardem *de facto* w modelowaniu i projektowaniu procesów zawierających istotne komponenty programowe. Jest powszechnie stosowany nie tylko dla modelowania procesów biznesowych. Zaimplementowano wiele narzędzi wspomagających wizualne projektowanie diagramów BPMN. Niestety, pomimo niezaprzeczalnych sukcesów semantyka BPMN i analiza własności logicznych stanowią ciągle słabe strony. Aby zapewnić niezawodną pracę systemów, należy przeprowadzić formalną analizę systemu.

Słowa kluczowe: Notacja Modelowania Procesów Biznesowych, BPMN, Reguły Biznesowe, Systemy Regułowe, Analiza Systemowa, Weryfikacja Systemów, Bezpieczeństwo, Niezawodność, Efektywność, Poprawność

Prof. Antoni Ligęza, PhD



Graduated from Faculty of Electrical Engineering, Automatics and Electronics (present: Faculty of Electrical Engineering, Automatics, Computer Science and Electronics, EAIiE), AGH University of Science and Technology in Cracow, Poland; received MSc in electronics/automatic control in 1980. After completing Doctors Studies he received his PhD degree in Computer Science (1983), and the habilitation (docent degree; Polish Dr habilitowany) in 1994 in Computer Science/Artificial Intelligence, both from the EAIiE Faculty at AGH. In 2006 he received the professor title from the President of Poland. His main research concern Knowledge Engineering (Artificial Intelligence) including knowledge representation and inference methods, rule-based systems, automated plan generation, technical diagnostics, logics and systems science. Some original research results include development of backward plan generation model (1983), independent discovery of dual resolution method for automated inference (1991), the concepts of granular sets and relations (2000), granular attributive logic (2003), diagnostic inference models in the form of logical AND/OR/NOT causal graphs (1995) and Potential Conflict Structures (1996). He was visiting reseracher/professor at Technical University of Denmark, Lyngby (1988), LAAS, Toulouse, France (1992, 1996), Universite de Nancy I, France (1994), University of Balearic Islands, Spain (1994, 1995, 2005), University of Girona, Spain (1996, 1997), and Universite de Caen, France (2004, 2005, 2007). Prof. Antoni Ligęza published over 200 research papers, including recent monograph Logical Foundations for Rule-Based Systems, Springer, 2006 (author), and Knowledge-Driven Computing, Springer, 2008 (co-editor and co-author). Member of IEEE Computer Society and ACM.

e-mail: ligeza@agh.edu.pl