A Hierarchical Approach for Configuring Business Processes

Mateusz Baran, Krzysztof Kluza,
Grzegorz J. Nalepa, Antoni Ligęza
AGH University of Science and Technology
al. A. Mickiewicza 30, 30-059 Krakow, Poland
E-mail: {matb,kluza,gjn,ligeza}@agh.edu.pl

***Abstract*—Business Process models in the case of real life systems are often very complex. Hierarchization allows for managing model complexity by "hiding" process details into sub-levels. This helps to avoid inconsistencies and fosters reuse of similar parts of models. Configuration, in turn, gives the opportunity to keep different models in one configurable model. In the paper, we propose an approach for configuring Business Processes that relies on hierarchization for more expressive power and simplicity. Our goal is achieved by allowing arbitrary n-to-m relationships between tasks in the merged processes. The approach preserves similar abstraction level of subprocesses in a hierarchy and allows a user to grasp the high-level flow of the merged processes.**

***Index Terms*—BPMN, Business Processes, Business Process Hierarchization, Business Process Configuration**

I. Introduction

ENTERPRISES take advantage of using Business Processes (BP) in their everyday practice. These processes define the way the company works by describing control flow between tasks. Design and development of such processes, especially more and more complex ones, require advanced methods and tools, e.g. [1], [2], [3].

Business Process Model and Notation (BPMN) [4] is a visual language used to model Business Processes. It is a set of graphical elements denoting such constructs as activities, splits and joins, events etc. (see Figure 1). These elements can be connected using control flow and provide a visual description of process logic [5]. Thus, a visual model is easier to understand than textual description and helps to manage software complexity [6].

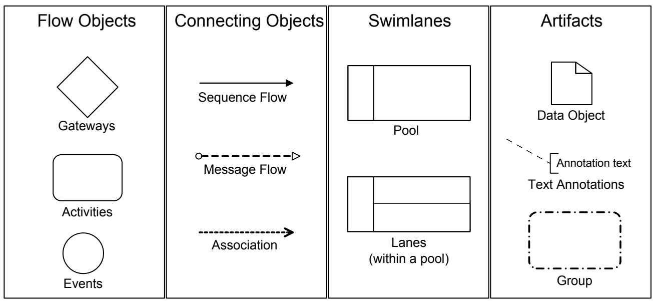

Figure 1. BPMN core objects

Although the BPMN notation is very rich when considering the number of elements and possible constructs, apart from the notation rules, some style directions for modelers are often used [7]. To deal with the actual BP complexity, analysts use various modularization techniques. Mostly, they benefit from their experience and modularize processes manually during the design because these techniques are not standardized. Modularization issue is important in the case of understandability of models [8]. Thus, guidelines for analysts, such as [9], emphasize the role of using a limited number of elements and decomposing a process model.

In the case of large collection of processes [10], especially modeled by different analysts, processes can be modularized in different ways on distinct granularity level [8]. Moreover, the processes in the collection can be similar [11], but this similarity is lost when the models are kept separately.

Automatic hierarchization and configuration, which is the subject of this paper, can help in preventing these problems. Hierarchization provides different abstraction levels and, properly developed, can ensure the same way of modularization for all the processes. Configuration gives the opportunity of unification of processes and enables to keep different models in one configurable model.

The rest of this paper is organized as follows: Section II presents motivation for our research. Section III and IV describe related works in the hierarchizaton and configuration areas. In Section V, we present our configuration BP approach which takes advantage of hierarchization. We evaluated the proposed approach based on the issue tracker case study. The paper is summarized in Section VI.

II. Motivation

Modern business applications require advanced modeling solutions to deal with the model complexity. Thus, several challenges in Business Process modeling can be distinguished:

- the *granularity modularization* challenge – how to model different processes similarly, especially in respect of abstraction layers [12].
- the *similarity capturing* challenge – it is not easy to grasp similarities in the collection of only partially similar models [13], especially if they are modularized differently.
- the *collection storing* challenge – how to store a large collection of models in some optimized way [14].

In our research, we deal with these challenges using automatic hierarchization that allows us to preserve similar abstraction level of subprocesses in a hierarchy. To address two other challenges, we propose a new BP configuration technique, that allows us to express similarities between different BP models in a simple, but comprehensive way. Our configuration is based on the hierarchical model prepared previously.

The aim of this paper is to present our approach for configuring Business Processes based on hierarchization. We present the automatic hierarchization algorithm that takes advantage of task taxonomy and the algorithm for configuration.

The proposed approach has several advantages. Thanks to the use of hierarchization, the obtained model structure incorporates similar activities, and the configuration step can be simplified. Thus, the configurable process diagram is easy to comprehend. Contrary to the existing configuration methods, in our approach we can bind not only one task with another, but also groups of tasks which were considered in hierarchization step. Such configuration technique address the abovementioned problems by managing both process model complexity and diversity.

III. Hierarchization Issues in Business Process Models

La Rosa et al. [15] distinguished 3 abstract syntax modifications that are related to modularization for managing process model complexity. These are:

1) vertical modularization – a pattern for decomposing a model into vertical modules, i.e. subprocesses, according to a hierarchical structure
2) horizontal modularization – a pattern for partitioning a model into peer modules, i.e. breaking down a model into smaller and more easily manageable parts, especially assigned to different users in order to facilitate collaboration.
3) orthogonal modularization – a pattern for decomposing a model along the crosscutting concerns of the modeling domain, such as security or privacy, which are scattered across several model elements or modules.

Our approach is consistent with the vertical and horizontal patterns. Although we decompose a model into subprocesses, in fact we use some additional information to decompose it, such as task assignment or task categories, which is an example of the second pattern instance. The last one, orthogonal pattern, requires to extend the notation, as in [16]; thus, it is not our case.

It is important to notice that such decomposition has several advantages:

- increases their understandability by "hiding" process details into sub-levels [8],
- decreases redundancy, helps avoid inconsistencies and fosters reuse by referring to a subprocess from several places [17], [18], [13], [19],
- decreases the error possibility [20],
- increases maintainability of such processes [9].

IV. Business Process Configuration

Business Process configuration is a tool for expressing similarities between different Business Process models. There are mechanisms for managing and comparing processes in large repositories [21], [22], refactoring of such repositories [14] as well as automatic extraction methods for cloned fragments in such process model repositories [14], [17]. However, our case differs from the existing approaches because we do not base on any directly visible similarity, but on previously defined taxonomy of states or roles in the processes etc. Moreover, our hierarchization algorithm forces the generation of similar models as a result.

There are a few methods of extraction of configurable processes. They focus on different goals. Analyzing digest configurable Business Process reveals high-level workflow that might not be apparent in particular models. The structure is partially lost in the process so this does not concern our approach.

The method of interest in this article are models merged into configurable model [23]. They allow the analyst to see several processes as special cases of one configurable model. The model emphasizes similarities preserving all the details.

Configurable Business Processes are also a good alternative to current reference model bases such as SAP. Instead of presenting the analyst a few example models, a more general, configurable solution can be delivered. It makes producing final models faster and less error-prone [24].

There is an active research field in the area of configurable Business Processes. In [25], Rosemann et al. describe an approach focused on hand-made diagrams for the purpose of reference modeling. La Rosa et al. [24] extend it with roles and objects.

Variant-rich process models were explored in PESOA project [26], [27]. They enable process designer to specify a few variants of a task.

Our approach is a specialized version of solution proposed by La Rosa in [28]. Hierarchization algorithm produces models of very specific structure and this fact is exploited in our approach.

V. Hierarchization-based Business Process Configuration Approach

We propose an approach for configuring Business Processes that relies on hierarchization for more expressive power and simplicity. The first goal is achieved by allowing arbitrary n-to-m relationships between tasks in merged processes. Taxonomy of tasks provides level of flexibility which many of current state-of-the-art solutions are lacking [25], [28].

Simplicity is the effect of constructing a very specific type of models during hierarchization. These models do not require general configuration algorithms that often produce complicated, hard to analyze diagrams. Sufficient configuration algorithm is described in Section V-C.

General flow of data in the whole approach is depicted in Figure 2.

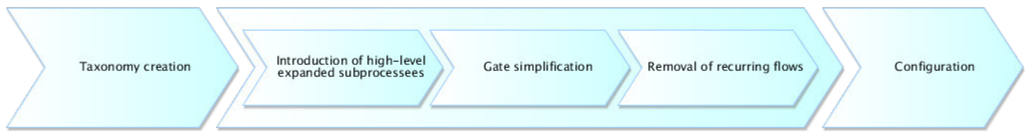
Taxonomy creation
Introduction of high-level expanded subprocessees
Gate simplification
Removal of recurring flows
Configuration

Figure 2. General flow of data in proposed approach

A. Case Study

To present our configuration approach, we chose 3 different BPMN models of bug tracking systems: Django[1], JIRA[2] and the model of the issue tracking approach in VersionOne[3].

A bug tracking system is a software application which helps in tracking and documenting the reported software bugs (or other software issues in a more general case). Such systems are often integrated with other software project management applications, such as in VersionOne, because they are valuable for the company.

Thus, apart from popularity, we selected such a case study because these kinds of processes have similar users assigned to similar kinds of tasks, the processes of different bug trackers present the existing variability, and such an example can be easily used to present our algorithm in a comprehensive way.

B. Automatic Hierarchization Algorithm

The hierarchization algorithm is given a BPMN model, a set of high-level BPMN tasks and an assignment of BPMN model's tasks to the high-level tasks. Using this information it constructs two-level hierarchical diagram. The lower level contains one diagram for each high-level task. Higher level diagram contains high level tasks (with lower-level diagram as subprocesses). This is done in such way to maximize simplicity and preserve semantics of original model.

Hierarchization is performed in several steps.

1) **Introduction of high-level expanded subprocesses**
In the first step, expanded subprocesses are introduced. Tasks are assigned to them according to given specification. Gateways are placed outside of all subprocesses unless all their incoming and outgoing flows lead to tasks of the same process. Intra-subprocess flows are kept. Inter-subprocess flows are replaced by:
   a) flow from source element to end event (in subprocess),
   b) OR-gateway right after subprocess (one per subprocess),
   c) flow from introduced gateway to target of initial flow with condition 'subprocess ended in event introduced in 1a'.

   This step is depicted in Figure 3. After this step is performed, assumption 1 of configuration algorithm (Section V-C) is fulfilled.

[1]See: https://code.djangoproject.com/
[2]See: http://www.atlassian.com/software/jira/
[3]See: http://www.versionone.com/

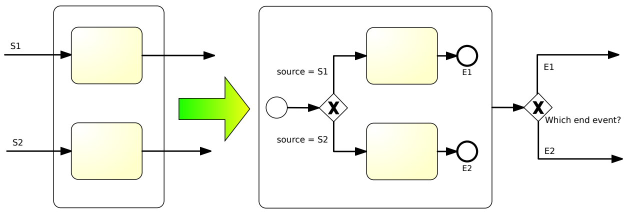

Figure 3. First step of hierarchization algorithm

2) **Gateway simplification**
The previous step introduced new gateways. The original diagram may contain unnecessary gateways too. Creating configurable diagram in proposed approach requires a very specific structure of high-level model. It can be achieved through gateway simplification.
The process of gateway simplification is depicted in Figure 4. In a simplified model one gateway G is placed after every subprocess $S(G)$ (unless it has only one outgoing flow which does not end in a gateway). Gateway G has outgoing flows to all subprocesses and end events reachable in original model from $S(G)$. Conditions labeling these flows are determined as follows.
Let $flow_N(G,T)$ and $flow_O(G,T)$ be the flows in the new and old graphs respectively from gateway G to target item T. Let $C_N(G,T)$ and $C_O(G,T)$ be the conditions on flow $flow_N(G,T)$ and $flow_O(G,T)$ respectively. Let $P(G,T)$ be the set of all paths in old graph (all gateways appear at most once) from G to P. Let $L(G)$ be the set of loops in gateway graph reachable from gateway G. Then the following hold:

$$all((G_1,G_2,\ldots,G_k),T) \text{ holds iff } C_O(G_k,T) \text{ and for all } i\in\{1,2,\ldots,k-1\} \text{ holds } C_O(G_i,G_{i+1})$$
$$C_N(G,T) \text{ holds iff exists } p\in P(G,T) \text{ such that } all(p,T)$$

Presented procedure works as long as graph of gateways is acyclic. Cycles need additional compensation for the fact that infinite looping is possible. We propose a solution where a new task "loop infinitely" is added and connected by flow from all gateways that allow looping. Condition on the new flow may be defined by analogy to the previous case:

$$all((G_1,G_2,\ldots,G_k)) \text{ holds iff } C_O(G_k,G_1) \text{ and for all } i\in\{1,2,\ldots,k-1\} \text{ holds } C_O(G_i,G_{i+1})$$
$$C_N(G,T) \text{ holds iff exists } p\in L(G) \text{ such that } all(p)$$

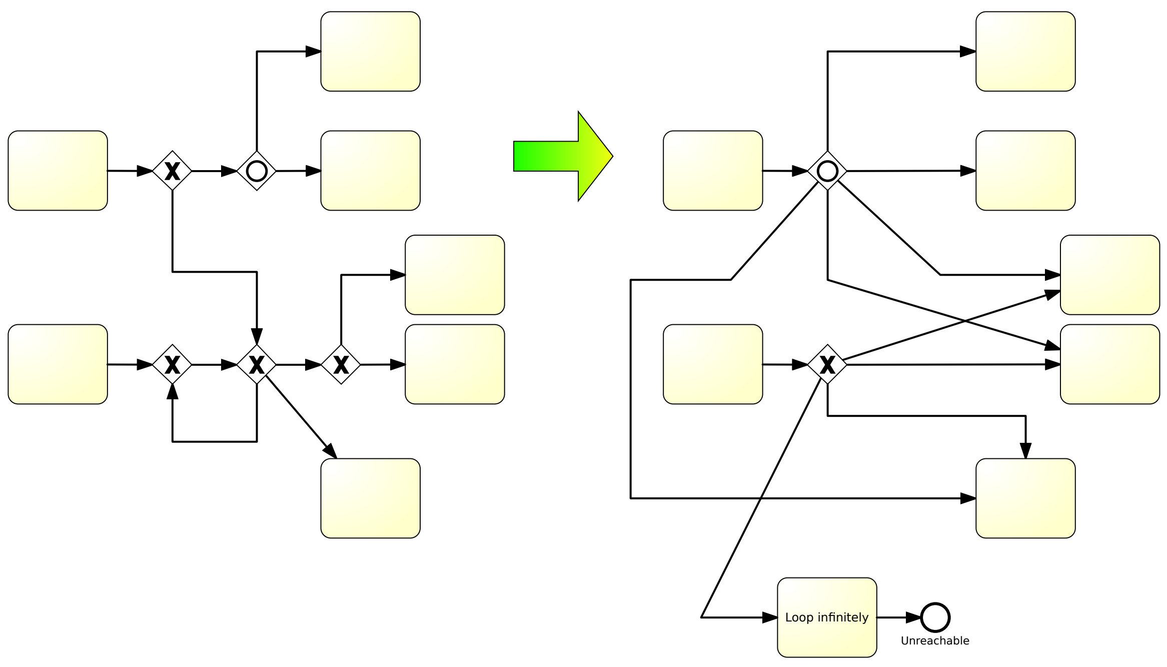

Figure 4. Second step of hierarchization algorithm (gateway simplification)

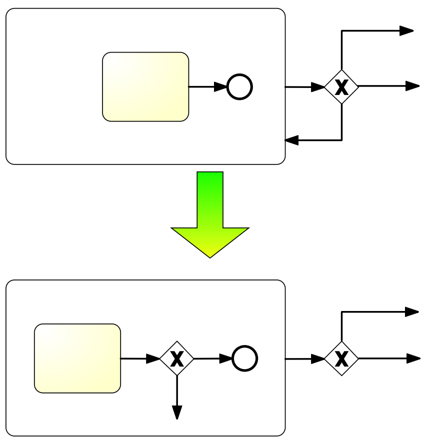

Figure 5. The idea of third step of hierarchization

Figure 4 shows a graph of gateways before and after simplification. Simplification assures that requirements 2 and 3 from Section V-C are fulfilled.

3) **Removal of recurring flows**
Last step of hierarchization is elimination of recurring flows. By this a flow from a gateway to activity preceding this gateway is meant (see Figure 5).
Before each end event in the subprocess that can result in recurring flow a new XOR gateway is placed. It has two output flows: one to end event and one to task or gateway a recurring flow would lead to (see Figure 5). The condition on the latter flow is created according to condition on original recurring flow.
This step of hierarchization algorithm makes requirement 4 of configuration process fulfilled. The result of hierarchization of Django issue tracking process can be seen in Figure 6.

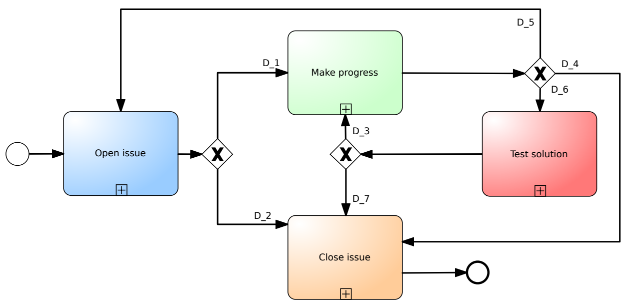

Figure 6. High-level diagram of Django issue tracking

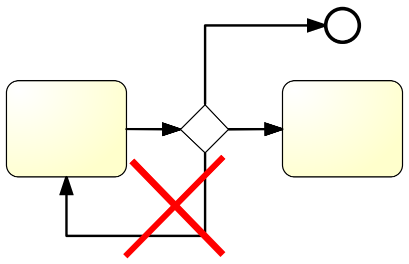

Figure 7. Possible flows to and from a gateway

C. Process Configuration

Let us be given N BPMN models such that:

1) all of them share the same set of tasks,
2) flows outgoing from tasks or start event end in a different task, end event or an (XOR or OR) gateway (Figure 7),
3) flows outgoing from gateways always end in tasks or an end event,
4) no flow outgoing from a gateway leads to a task that has a flow to this gateway.

Then the configurable model that entails all the given models can be defined as follows:

1) configurable diagram has one start event, all the specified tasks and all the end events from N given models,
2) for all tasks and start event (let i be the current item):
   a) If all diagrams have flow outgoing from i that ends in (the same) task or the only end event then the same flow exists in merged diagram.
   b) If in at least one model the flow f ends in a gateway, the merged model has a configurable gateway after i. It is a configurable type gateway if there are diagrams with two different types of gateways (or one without gateway).
   c) If and only if any input diagram gateway after i has a flow to an item, the configurable gateway has a flow to this item too. The flows are labeled with model number and condition from that model.

D. Approach Evaluation

As we tested our approach on the three issue tracking systems, the results we got are optimistic. The obtained model is simple and comprehensible. Figure 9 compares initial and hierarchical versions of Django system. The three hierarchized models, simplified by the algorithm, can be simultaneously compared on high level and on the subprocess level. The final high level configurable model is presented in Figure 8.

One of the drawbacks of our approach is that conditions on control flows outgoing from gateways may become complex after hierarchization. However, it is not an obstacle in understanding of high level flow in the process, which is the goal of the approach.

VI. Conclusion and Future Work

The research presented in this paper addresses three challenges in Business Process modeling, which we distinguished in Section II. These are granularity modularization, similarity capturing and collection storing challenges.

In the paper, we proposed automatic hierarchization algorithm that takes advantage of task taxonomy and allows us to preserve similar abstraction level of subprocesses in a hierarchy. A Business Process configuration technique, based on the hierarchization result is presented as well. It allows for expressing similarities between different BP models in a simple but comprehensive way. Thanks to this, a user can grasp the high-level flow of the merged processes.

In comparison to other approaches, our hierarchization algorithm supports arbitrary n-to-m relationships between tasks in the merged processes.

To get a proof of concept of our approach, we narrowed our attention to the subset of BPMN, similarly expressive to EPC. Thanks to the use of the taxonomy shared by the three models and the hierarchization algorithm, the configuration approach is straightforward.

In future work, we consider to extend the approach in several ways, e.g. to allow more BPMN elements or multi-level diagrams [1], and to integrate it with Business Rules [29], especially in the XTT2 representation [30], [31], in order to use control flow as inference flow [32] and to allow for automatic verification of models [33], [34], [35]. Moreover, automatic generation of taxonomy using some process metrics [11], [36] is also considered, as well as automatic assignment of tasks to subprocesses based on Natural Language Processing.

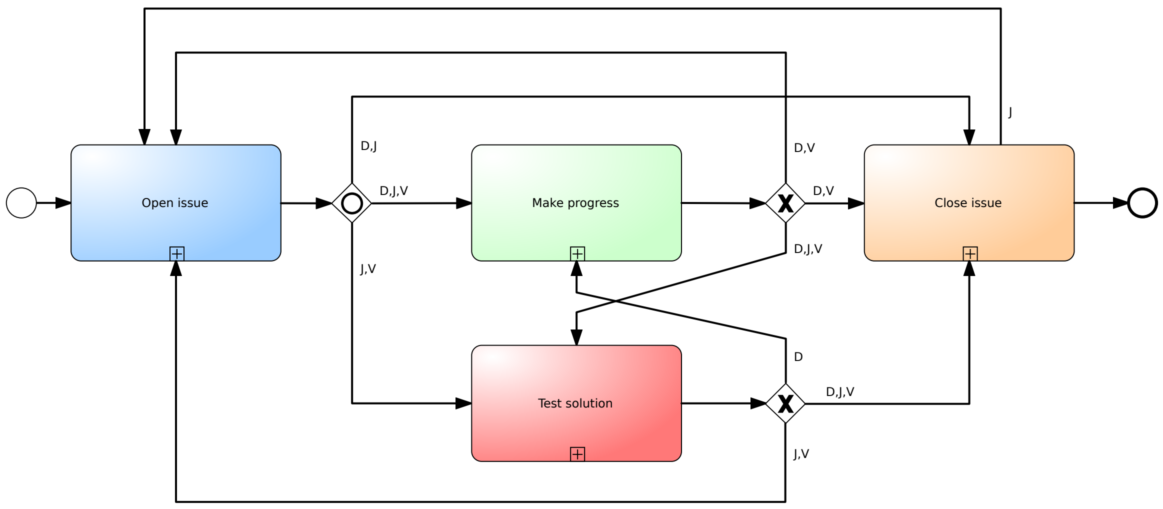

Figure 8. Result of the proposed algorithm (after configuration)

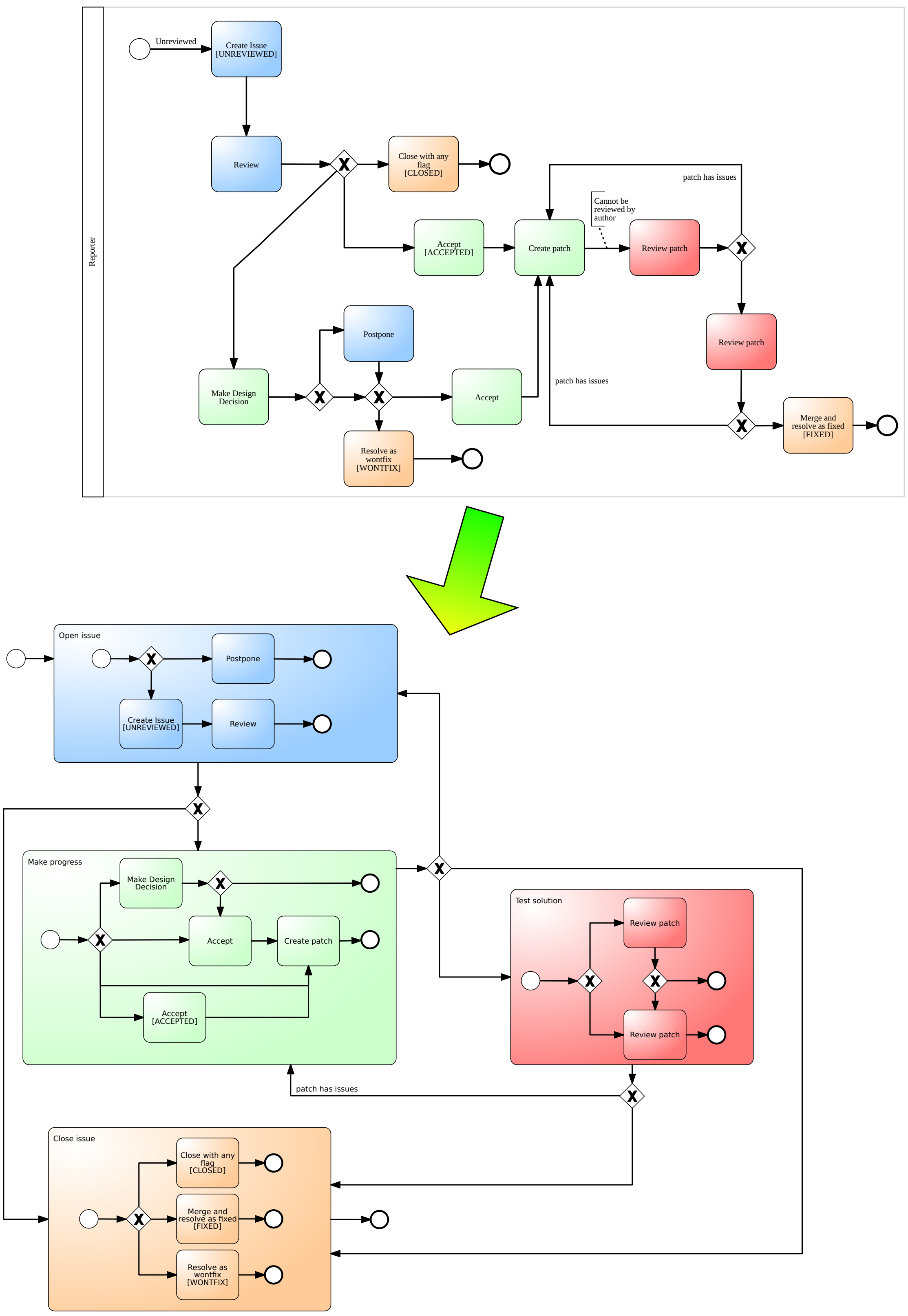

Figure 9. Comparison of initial diagram and its hierarchical version

ACKNOWLEDGMENT

The paper is supported by the AGH UST Grant.

REFERENCES

[1] K. Kluza, K. Kaczor, and G. J. Nalepa, "Enriching business processes with rules using the Oryx BPMN editor," in *Artificial Intelligence and Soft Computing: 11th International Conference, ICAISC 2012: Zakopane, Poland, April 29–May 3, 2012*, ser. Lecture Notes in Artificial Intelligence, L. Rutkowski and [et al.], Eds., vol. 7268. Springer, 2012, pp. 573–581.

[2] K. Kaczor, G. J. Nalepa, Ł. Łysik, and K. Kluza, "Visual design of Drools rule bases using the XTT2 method," in *Semantic Methods for Knowledge Management and Communication*, ser. Studies in Computational Intelligence, R. Katarzyniak, T.-F. Chiu, C.-F. Hong, and N. Nguyen, Eds. Springer-Verlag, 2011, vol. 381, pp. 57–66, DOI: 10.1007/978-3-642-23418-7. [Online]. Available: http://www.springerlink.com/content/h544g4238716m320/

[3] M. Szpyrka, "Exclusion rule-based systems – case study," in *International Multiconference on Computer Science and Information Technology*, vol. 3, Wisła, Poland, October 20-22 2008, pp. 237–242.

[4] OMG, "Business Process Model and Notation (BPMN): Version 2.0 specification," Object Management Group, Tech. Rep. formal/2011-01-03, January 2011.

[5] T. Allweyer, *BPMN 2.0. Introduction to the Standard for Business Process Modeling*. Norderstedt: BoD, 2010.

[6] G. J. Nalepa and K. Kluza, "UML representation for rule-based application models with XTT2-based business rules," *International Journal of Software Engineering and Knowledge Engineering (IJSEKE)*, vol. 22, no. 4, pp. 485–524, 2012.

[7] B. Silver, *BPMN Method and Style*. Cody-Cassidy Press, 2009.

[8] H. Reijers, J. Mendling, and R. Dijkman, "Human and automatic modularizations of process models to enhance their comprehension," *Information Systems*, vol. 36, no. 5, pp. 881–897, 2011.

[9] J. Mendling, H. A. Reijers, and W. M. P. van der Aalst, "Seven process modeling guidelines (7pmg)," *Information & Software Technology*, vol. 52, no. 2, pp. 127–136, Feb 2010.

[10] Z. Yan, R. Dijkman, and P. Grefen, "Business process model repositories – framework and survey," *Information and Software Technology*, vol. 54, no. 4, pp. 380–395, 2012.

[11] R. Dijkman, M. Dumas, B. van Dongen, R. Käärik, and J. Mendling, "Similarity of business process models: Metrics and evaluation," *Information Systems*, vol. 36, no. 2, pp. 498–516, Apr 2011.

[12] D. V. Nuffel and M. D. Backer, "Multi-abstraction layered business process modeling," *Computers in Industry*, vol. 63, no. 2, pp. 131–147, 2012.

[13] F. Pittke, H. Leopold, J. Mendling, and G. Tamm, "Enabling reuse of process models through the detection of similar process parts," in *Business Process Management Workshops*, ser. Lecture Notes in Business Information Processing, M. Rosa and P. Soffer, Eds. Springer Berlin Heidelberg, 2013, vol. 132, pp. 586–597.

[14] B. Weber, M. Reichert, J. Mendling, and H. A. Reijers, "Refactoring large process model repositories," *Computers in Industry*, vol. 62, no. 5, pp. 467–486, 2011.

[15] M. La Rosa, P. Wohed, J. Mendling, A. ter Hofstede, H. Reijers, and W. M. P. Van der Aalst, "Managing process model complexity via abstract syntax modifications," *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 4, pp. 614–629, 2011.

[16] C. Cappelli, J. C. Leite, T. Batista, and L. Silva, "An aspect-oriented approach to business process modeling," in *Proceedings of the 15th workshop on Early aspects*, ser. EA '09. New York, NY, USA: ACM, 2009, pp. 7–12.

[17] R. Uba, M. Dumas, L. García-Bañuelos, and M. Rosa, "Clone detection in repositories of business process models," in *Business Process Management*, ser. Lecture Notes in Computer Science, S. Rinderle-Ma, F. Toumani, and K. Wolf, Eds. Springer Berlin Heidelberg, 2011, vol. 6896, pp. 248–264.

[18] M. Dumas, L. García-Bañuelos, M. L. Rosa, and R. Uba, "Fast detection of exact clones in business process model repositories," *Information Systems*, vol. 38, no. 4, pp. 619–633, 2013.

[19] N. Zaaboub Haddar, L. Makni, and H. Ben Abdallah, "Literature review of reuse in business process modeling," *Software & Systems Modeling*, pp. 1–15, 2012.

[20] J. Mendling, G. Neumann, and W. Aalst, "Understanding the occurrence of errors in process models based on metrics," in *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, ser. Lecture Notes in Computer Science, R. Meersman and Z. Tari, Eds. Springer Berlin Heidelberg, 2007, vol. 4803, pp. 113–130.

[21] M. Kunze and M. Weske, "Metric trees for efficient similarity search in large process model repositories," in *Business Process Management Workshops*, ser. Lecture Notes in Business Information Processing, M. Muehlen and J. Su, Eds. Springer Berlin Heidelberg, 2011, vol. 66, pp. 535–546.

[22] R. Dijkman, M. L. Rosa, and H. A. Reijers, "Managing large collections of business process models – current techniques and challenges," *Computers in Industry*, vol. 63, no. 2, pp. 91–97, 2012.

[23] W. M. van der Aalst, "Business process management: A comprehensive survey," *ISRN Software Engineering*, vol. 2013, 2013.

[24] M. L. Rosa, M. Dumas, A. H. ter Hofstede, and J. Mendling, "Configurable multi-perspective business process models," *Information Systems*, vol. 36, no. 2, pp. 313 – 340, 2011.

[25] M. Rosemann and W. M. P. van der Aalst, "A configurable reference modelling language," *Inf. Syst.*, vol. 32, no. 1, pp. 1–23, Mar. 2007.

[26] F. Puhlmann, A. Schnieders, J. Weiland, and M. Weske, "Variability Mechanisms for Process Models. PESOA-Report TR 17/2005, Process Family Engineering in Service-Oriented Applications (pesoa). BMBF-Project."

[27] A. Schnieders and F. Puhlmann, "Variability mechanisms in e-business process families," in *Proc. International Conference on Business Information Systems (BIS 2006*, 2006, pp. 583–601.

[28] M. L. Rosa, M. Dumas, R. Uba, and R. M. Dijkman, "Business process model merging : An approach to business process consolidation," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 22, no. 2, 2013.

[29] G. J. Nalepa, "Proposal of business process and rules modeling with the XTT method," in *Symbolic and numeric algorithms for scientific computing, 2007. SYNASC Ninth international symposium. September 26–29*, V. Negru and et al., Eds., IEEE Computer Society. Los Alamitos, California ; Washington ; Tokyo: IEEE, CPS Conference Publishing Service, september 2007, pp. 500–506.

[30] G. J. Nalepa, A. Ligęza, and K. Kaczor, "Formalization and modeling of rules using the XTT2 method," *International Journal on Artificial Intelligence Tools*, vol. 20, no. 6, pp. 1107–1125, 2011.

[31] A. Ligęza and G. J. Nalepa, "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*, vol. 1, no. 2, pp. 117–137, 2011.

[32] G. Nalepa, S. Bobek, A. Ligęza, and K. Kaczor, "Algorithms for rule inference in modularized rule bases," in *Rule-Based Reasoning, Programming, and Applications*, ser. Lecture Notes in Computer Science, N. Bassiliades, G. Governatori, and A. Paschke, Eds., vol. 6826. Springer Berlin / Heidelberg, 2011, pp. 305–312.

[33] K. Kluza, T. Maślanka, G. J. Nalepa, and A. Ligęza, "Proposal of representing BPMN diagrams with XTT2-based business rules," in *Intelligent Distributed Computing V. Proceedings of the 5th International Symposium on Intelligent Distributed Computing – IDC 2011, Delft, the Netherlands – October 2011*, ser. Studies in Computational Intelligence, F. M. Brazier, K. Nieuwenhuis, G. Pavlin, M. Warnier, and C. Badica, Eds. Springer-Verlag, 2011, vol. 382, pp. 243–248.

[34] M. Szpyrka, G. J. Nalepa, A. Ligęza, and K. Kluza, "Proposal of formal verification of selected BPMN models with Alvis modeling language," in *Intelligent Distributed Computing V. Proceedings of the 5th International Symposium on Intelligent Distributed Computing – IDC 2011, Delft, the Netherlands – October 2011*, ser. Studies in Computational Intelligence, F. M. Brazier, K. Nieuwenhuis, G. Pavlin, M. Warnier, and C. Badica, Eds. Springer-Verlag, 2011, vol. 382, pp. 249–255.

[35] F. Coenen *et al*, "Validation and verification of knowledge-based systems: report on eurovav99," *The Knowledge Engineering Review*, vol. 15, no. 2, pp. 187–196, 2000.

[36] K. Kluza and G. J. Nalepa, "Proposal of square metrics for measuring business process model complexity," in *Proceedings of the Federated Conference on Computer Science and Information Systems – FedCSIS 2012, Wroclaw, Poland, 9-12 September 2012*, M. Ganzha, L. A. Maciaszek, and M. Paprzycki, Eds., 2012, pp. 919–922.