RESEARCH PAPERS OF WROCŁAW UNIVERSITY OF ECONOMICS No. 147
Advanced Information Technologies for Management – AITM 2010 2010

Radosław Klimek, Paweł Skrzyński, Michał Turek
AGH University of Science and Technology, Kraków, Poland
e-mail: {rklimek;skrzynia;mitu}@agh.edu.pl

DEDUCTION BASED VERIFICATION OF BUSINESS MODELS

Abstract: The paper presents a formal verification of the business processes expressed in BPMN. Verification is based on deductive reasoning. Automatic transformations of basic BPMN workflow patterns to temporal logic formulae are introduced. These formulae constitute a system specification and they are later processed using semantic tableaux method. In general, such reasoning technique has many advantages over the traditional approach, i.e., the resolution method. The paper provides automatic transformations for five basic BPMN workflow patterns and the example process is provided with description in BPMN diagram. The related temporal logic formula is obtained through automatic transformations, then the algorithm of reasoning using semantic tableaux methodology is applied to verify the business model.

Keywords: deductive reasoning, business processes, BPMN, workflow design patterns, formal verification, semantic tableaux.

1. Introduction

Business Process Modeling Notation (BPMN) is becoming more and more popular and is widely used to describe business processes running in any given company. It might also be treated as an additional layer providing graphical description of the processes which are implemented in the languages such as BPEL (Business Process Execution Languages), which enables execution of the processes on the workflow engines. BPMN and BPEL are compliant with the Service-Oriented Architecture (SOA) software design which is a business-centric IT architectural approach that supports integrating any given business as linked, repeatable business tasks, or services.

UML has also become an industrial standard and is widely used as a general purpose modeling language. It can also be used to model business processes. UML provides thirteen different types of diagrams which might be used to describe system from different perspectives, whereas some of these diagrams may be used in business modeling. Despite the increasing popularity of BPMN and UML it seems there is no formal mechanism provided in the modeling phase other than process simulation which is not equal to formal verification.

The paper introduces the deductive reasoning about business models, which allows to examine the properties of the model. Presented framework is based on an innovative approach which brings added value to at least two points. The first is the automatic generating of a logical specification of a bussiness model. Manual creation of such a specification is difficult and questionable because of the need to tediously create a huge number of logical formulae for which it is uncertain whether and where an error was made. And for this reason, rules generation of temporal logic formulae for BPMN's design patterns are introduced. The second important point is the application of the inference method which is based on semantic tableaux approach. This approach is in some opposition to the resolution method and its advantage is receiving at each step of reasoning procedure simpler formulae by their decomposition. Semantic tableaux method is goal-oriented, which means that the method operates directly on the goal (formula) to be proved. Another advantage of the tableaux method is the ability to identify inconsistencies in the inference tree. Since the specifications provided by OMG implement the Model Driven Architecture, it is reasonable to provide additional chain in the software development chain which is responsible for the verification of the system. So, summing up, the originality of the method is based on these two issues, namely the automatic generation of the logical layer specifications and and the non-standard methods of reasoning for business models.

While modeling business processes with BPMN the development chain starts with creating initial drafts of the business processes, which in most cases is performed by providing informal textual specifications usually existing in the company. The Business Process Modeling Notation (BPMN) is a standard notation to provide more formal description of the business processes as an easily understandable set of diagrams. The advantage of the BPMN is its simplicity – it is easy to use and understand by business analysts who create initial drafts of processes, technical developers responsible for implementing those processes and finally by the business people who will manage and monitor those processes. The important fact is that the BPMN diagrams might be mapped to execution languages, such as BPEL, and further deployed on the Workflow Engines and then executed.

During modeling systems with UML the modeling process usually starts with constructing use case diagrams which show system functionalities from the external actor point of view. The diagram does not provide information about the ways in which the goals are achieved but what has to be achieved. Each use case comes with a scenario which defines the set of activities performed within a use case. A common way of describing scenario is by using an activity diagram which describes control flow from one activity to another. The diagram does not provide information on how the involved objets collaborate with each other or the way they act.

It seams that it is reasonable to apply the formalism based on temporal logic and semantic tableaux before executing[1] those models in order to verify them. This does not ensure addressing all the problems. The approach described in the paper allows formal verification of the model or business model at modeling phase. Moreover, it might be performed automatically. The problem that might be addressed by applying such formalism might be defined as answers to the following issues:

– effective use of the deductive approach to reasoning about the model/business model,
– automation of the process of inference,
– generating temporal logic formulae from the business model expressed in BPMN or use cases described by scenarios in activity diagrams,
– contradictions existence in the model/business model,
– all activities in the model/business model achievable.

2. Business processes

Business Process Modeling Notation BPMN is a standard graphical notation provided by the Business Process Management Initiative (BPMI) for the modeling of business processes. The main goal of BPMN is to provide a notation which is understandable by all business users, from business analysts who create the initial drafts of the processes, to technical developers responsible for implementing the technology that will enable execution of the processes, and finally, to business people who will manage and monitor those processes [*Business Process*... 2009]. Thus, BPMN creates a standardized bridge for the gap between the business process design and process implementation.

While implementing Model Driven Architecture in the software development process we should aim at automatic transformations between steps. One of the key transformations performed in such process is the BPMN (modeling) to BPEL (execution) transformation. The problem is that there are some limitations in executing processes modeled in BPMN which are caused by semantic limitations as expressed in BPMN FAQ: "By design there are some limitations to the process topologies that can be described in BPEL, so it is possible to represent processes in BPMN that cannot be mapped to BPEL". Those limitations should be also taken into consideration while verifying the business model. The question is how to overcome those limitations? It appears that by using workflow design patterns we can describe any business model in such a detailed way as required. Those design patterns are BPMN representations of some of the most frequently used workflow patterns as identified by the Workflow Pattern Initiative (www.workflowpatterns.com). The aim of the workflow pattern initiative is to: "Provide a conceptual basis for process

[1] Executing BPMN models might be performed by transforming them to BPEL, while modeling with UML it requires some detailed modeling using other types of diagrams as well.

technology. In particular, the research provides a thorough examination of various perspectives (control flows, data, resource and exception handling) that need to be supported by a workflow language or business process modeling language". Hence it is sufficient to provide automatic transformations of the patterns to temporal logic formulae to enable formal verification.

Another goal is to ensure that XML languages designed for the execution of business processes, such as BPEL4WS (*Business Process Execution Language for Web Services*), can be visualized with a business-oriented notation since business analysts feel very comfortable with visualizing business processes, reflecting the way companies work, in a flow-chart format. On the other hand there has been much activity in the recent years in developing Web service-based XML execution languages for Business Process Management (BPM) systems. As a result of these efforts languages such as as BPEL (Business Process Execution Language) provide a formal mechanism for the definition of business processes. The key element of such languages is that they are optimized for the operation and inter-operation of BPM Systems. The optimization of these languages for software operations renders them less suited for direct use by humans to design, manage, and monitor business processes. BPEL4WS has both graph and block structures and utilizes the principles of formal mathematical models, such as π-calculus [*Business Process*... 2009]. This provides the foundation for business process execution to handle the complex nature of internal and B2B interactions to take advantage of the benefits of Web services. Given the nature of BPEL, a complex business process could be organized in a potentially complex, disjointed, and unintuitive format that is handled very well by a software system (or a computer programmer), but would be hard to understand by the business analysts and managers tasked to develop, manage, and monitor the process. Thus, there is a human level of "inter-operability" or "portability" that is not addressed by these Web service-based XML execution languages. Hence this leads to a technical gap between the format of the initial design of business processes and the format of the languages, such as BPEL, that will execute these business processes. The gap is bridged with BPMN – a formal mechanism that maps the appropriate visualization of the business processes (a notation) to the appropriate execution format (a BPM execution language) for these business processes provided.

2.1. BPMN notation symbols

Standardizing graphical notation for business process description should facilitate the understanding of the performance collaborations and business transactions within and between the organizations. Such approach should ensure that businesses will understand themselves and participants in their business and will enable organizations to adjust to new internal and B2B business circumstances quickly. To achieve this goal BPMN follows the tradition of flowcharting notations for readability and still provides a mapping to the executable constructs. BPMN is using the experience of the business process notations which have preceded BPMN to create the next generation notation that combines readability, flexibility, and expandability. BPMN also enhances the capabilities of traditional business process notations by inherently handling B2B business process concepts, such as public and private processes and choreographies, together with advanced modeling concepts, such as exception handling, transactions, and compensation.

One of the main requirements for the BPMN were simplicity of creation business process models while at the same time providing ability to handle the complexity inherent to business processes. The approach taken to handle those conflicting requirements was to organize the graphical aspects of the notation into specific categories resulting in providing a small set of notation categories so that the reader of a BPMN diagram can easily recognize the basic types of elements and understand the diagram. Within the basic categories of elements, additional variation and information can be added to support the requirements for complexity without changing the basic look and feel of the diagram. The four basic categories of elements are:

1. Flow Objects: events, activities, gateways.
2. Connecting Objects: sequence flow, message flow, association.
3. Swimlanes – used for grouping objects: pools and lanes.
4. Artifacts: data objects, groups, annotations.

The look and feel of those elements is intuitive and based on the other popular notations such as UML activity diagrams. Basic shapes are identical with the ones introduced by UML while BPMN offers larger number of "control flow constructs" (i.e. gateways). BPMN also offers a predefined set of event types, whereas UML activity diagrams do not specify any signals with special semantics a priori (they are all user-defined). However, the goal of the paper is not to compare those notations. As a conclusion we might say that everyone who is familiar with UML or even other flow chart diagrams will easily understand any given BPMN diagram.

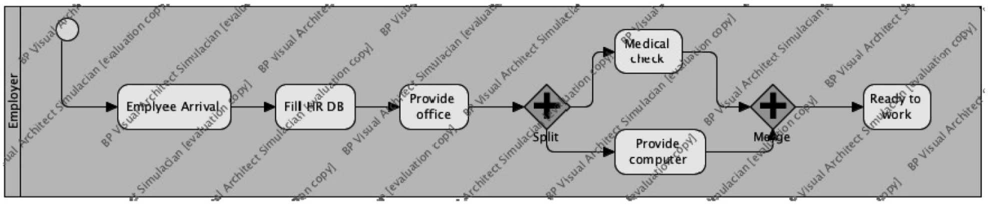

Figure 1. Simple process description in BPMN

Figure 1 describes the process of employment of a new employee (which might be modeled in a different way depending on a company). When a new employee arrives at a company, first a record needs to be created in the Human Resources database, after that an office has to be provided. As soon as those two activities have been completed, the employee has to undergo a medical check-up. During that time, a computer is provided. This is only possible when both an office has been set up and an account created in the information system from the human resource database. When both the computer and the medical check-up have been performed, the employee is ready to work.

2.2. Design patterns

Riehle and Zullighoven in their work [1996] described the *patterns* as: "the abstraction from a concrete form which keeps reccurring in specific non-arbitrary contexts". The process patterns are examples which show how to connect activities together to solve a common problem. Similar to the approach known from the object-oriented design (Gamma et al. [1995] first catalogued systematically 23 patterns in object oriented systems), process design pattern identification and documentation was well served by the article by van der Aalst et al. [2003]. Gradually building in complexity, process patterns were broken down into six categories:

1. Basic control flow patterns.
2. Advanced branching.
3. Structural.
4. Multiple instances.
5. State based.
6. Cancellation.

In this paper the considerations will be limited to the basic control flow patterns which are listed below.

Sequence – an activity in a workflow process is enabled after the completion of another activity in the same process.

Parallel Split – a point in the workflow process where a single thread of control splits into multiple threads of control which can be executed in parallel, thus allowing activities to be executed simultaneously or in an order.

Synchronization – a point in the workflow process where multiple parallel subprocesses/activities converge into one single thread of control, thus synchronizing multiple threads with the assumption that each incoming branch of synchronizer is executed only once.

Exclusive choice – a point in the workflow process where, based on the decision or workflow control data, one or several branches are chosen.

Simple Merge – a point in the workflow process where two or more alternative branches come together without synchronization with the assumption that none of the alternative branches is ever executed in parallel.

In the further part of the paper, transformations of each pattern given above to the logic formula, which is processed using semantic tableaux methodology, are introduced. Using these five basic workflow patterns simple processes might be modeled as presented on Figure 1. The process described on this chart uses three out of five patterns described above:

1. Sequence: filling record in Human Resources database takes place after the activity “Employee arrival” is finished. Office is provided (activity “Provide office”) as soon as the record in HR database has been created.

2. Parallel split: medical check-up is done simultaneously with providing office.

3. Synchronization: after a medical check is completed and a computer is provided, an employee is ready to work.

In the further part of the paper we will provide automatic translation to logical formulae for each of the five workflow patterns and we will build logical formula for the example process by combining the formulae generated from the related three workflow patterns used in the example.

2.3. UML modeling – use cases and scenarios

As mentioned above, the first step performed in modeling system with UML is usually to construct use case diagram describing functional requirements. The definition of the use case is as follows: the sequence of the transactions within the system which is aimed at providing measurable value to the system client [Booch et al. 1998]. These diagrams do not provide any information about how those functionalities are achieved hence the next step is to provide scenarios for the use cases. Such scenario is the main part of the use case since it contains all the business logic. The popular way to provide scenarios is by textual (or tabular) descriptions containing the following sections: actors involved, initial conditions, end conditions, main actions sequence, alternative actions sequence (if exists, might be many). Such description is an input for constructing *activity diagrams* which of course could not be created automatically but the authors provide some hints which simplify this process. First of all we should consider actions and actors – which actions are performed by which actors. Secondly we should consider the events which activate the steps of the scenario. Finally we should search for complex activities which might require separate diagram. While connecting activities we should consider:

– the order in which they are executed,
– actions that take place concurrently,
– where to fork and where to join,
– if the termination of the activity allows the scenario to continue.

As mentioned above, a complex process of creating activity diagram cannot be automated since the textual description of the scenario is rather informal. However, simplification of such process might be achieved by applying the following algorithm [Klimek et al. 2010]:

1. Create global activity list (1) related to the use case diagram and assign each activity to an actor.

2. For each activity from the list (1) create elementary operations list (2). Put the lists (2) on activity diagrams connecting the head with initial symbol and the tail with final symbol.

3. For each list item (1) create the set of lists for alternative scenarios (3).

4. Create events (4) and conditions (5) affecting the the choice of alternative scenario (3).

5. Create list of relations between use cases (6) («include», «extend»).

6. Basing on list (6) merge related lists (2) using fork symbol for «include» stereotype. Add joins basing on the information provided by the scenario for the use case related to the list (2). For the stereotype «extend» use decision symbol and join flows basing on the information provided by the scenario for the uses case related to the list (2).

7. On the basis of lists (4) and (5) add all the lists (3) to the related lists (2) on positions defined by the use case scenario. While merging use join symbol. If the the description does not allow joining use final symbol for the list (3). If the join location exists, connect the tail of the each list (3) with the related list (2) using a flow symbol.

Properly constructed activity diagram should:

– describe all the possibilities (the flow must not stuck),
– join-forks should be balanced – the number branches at fork symbol must be equal to the number of branches at related join symbol,
– define entry point and end point for each action clearly defining the direction in which the flow must continue,
– describe in its structure use case relations («extend», «include») and the alternative scenarios.

The paper of Klimek et al. [2010] contains an example of activity diagram's extraction of formulae and inferencing about model's properties using semantic tableaux method.

3. A deductive apparatus

Formal reasoning about the system plays a key role in the system development because it enables reliable verification of desired system properties. There are basically two approaches to formal verification of information systems [Clarke et al. 1996]. The first one is based on states' exploration and the second one on deductive inference. In both cases the importance of *temporal logic* [Emerson 1990; Klimek 1999] which is a convenient formalism for specification and verification of sequences of events without a strict timing, should be emphasised. Temporal logic formulae can easily express liveliness and safety properties which play a key role in proving properties of the system. Construction of a deductive system must take into account a number of issues, including the structure of time domain, and it is a fairly complex process. However, axiomatic and deductive system for the *smallest temporal logic* is defined, [Van Benthem 1995], as a classical propositional calculus' extension of axiom $\Box(P \Rightarrow Q) \Rightarrow (\Box P \Rightarrow \Box Q)$ and inference rule $|-P \Rightarrow |-\Box P$. We focus on linear time temporal logic as it is sufficient to define most of system properties.

These two approaches, i.e. states' exploration and theorem proving, are well established, however, during recent years there was a particularly significant progress in the field of states' exploration and there are already available software packages for model checking. The inference method is now behind states' exploration. It seems that there are two reasons for this situation. The first is a problem of choosing a deductive system itself, i.e. a method of inference. The second is a method to specify, or to obtain, a system requirements as a set of temporal logic formulae, in particular the automation of this process. In general, it is not obvious how to generate a large number of formulae which specify the model. However, it seems that the BPMN models are convenient for such a process of automatic or partly automatic extraction of formulae. This follows the nature of BPMN models which constitute a kind of logical network. The network consists of a flow control in which data flows are minimized. Another important issue is the method of automatic reasoning. Inference based on traditional methods, i.e. axiomatization and resolution, is of course possible but much more interesting alternative seems to be the method of *semantic tableaux*. The method is well known in classical logic but it also undergoes automation and can be applied in the field of modal logics [*Handbook of Tableau...* 1999; Wolper 1985]. It seems that it has some advantages over the traditional axiomatic approach. Broadly speaking, the method is based on the formula decomposition. At each step of well-defined procedure, we obtain formulae which are far simpler while the logical connectives are removed. At the end of the decomposition procedure we search for contradictions in all branches of the received tree. As already mentioned, this method has some advantages. In the classical approach, starting from axioms, longer and more complicated formulae are generated and derived. Formulae are getting longer and longer with every step, and only one of them will lead to the verified formula. The method of semantic tableaux is characterized by the reverse strategy. Though we start with a long and complex formula, it becomes less complex and shorter with every step of the decomposition procedure. In addition, this method provides, through the so-called open branches of the semantic tree, information about the source of an error, if it is found.

The proposed system of the logical and automatic reasoning about properties of the system (business model) for the BPMN environment is presented in Figure 2. The system consists of two basic parts. The first is responsible for the creation of a system specification. This is done in such a way that on the basis of obtained graphical BPMN model temporal logic formulae are generated. These formulae represent properties of the system and they constitute its specification. Formula generation is done by the extraction directly from the network activities expressed in the BPMN language. Extraction is focused on the BPMN patterns and it is shown in the further part of this work. Formulae are collected in a module that stores the specification of the system. Obtained formulae are treated as a set of system properties. Due to the further inference about the system, we consider it as a conjunction of individual properties $p_1 \wedge ... \wedge p_n \equiv P$. The second part of the system is responsible for the creation of the desired properties of the system. It is expressed in temporal logic formulae, too. The easiest way to obtain such a formula is a (manual) identification of formula which describes property Q using any formula editor for temporal logic. Formulae are stored in a separate module that stores the verified properties of the system. Another way to obtain a formula is query language BPQL (Business Process Query Language). This way of obtaining the formulae will not be examined in this paper. It rather represents the next step in research and has been mentioned here in order to obtain a more general perspective. Both the specification of the system and its properties are input to the module executing automated reasoning in temporal logic using semantic tableaux method, i.e. *Semantic Tableaux Temporal Prover*. This is done in such a way that the formula $P \Rightarrow Q$, or, more accurate:

$$p_1 \wedge ... \wedge p_n \Rightarrow Q \tag{1}$$

after its negation, is placed at the root of the inference tree. Then the formula is decomposed by well-defined rules of the method. Finding a contradiction in all branches of the tree means no valuation satisfies a formula placed in the root. When all branches of the tree have contradictions, it means that the inference tree is *closed*. This consequently leads to the statement that the initial formula 1 is true.

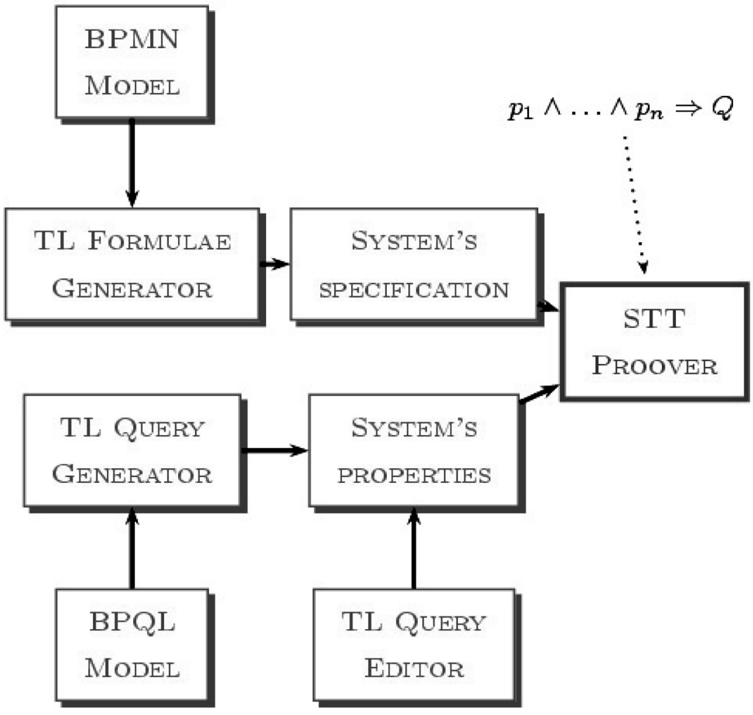

Figure 2. Model of a semantic tableaux deduction system for business processes

Let us summarize the entire algorithm due to the system shown in Figure 2:

1) generate in an automatic way a specification of the system, which is then stored as a conjunction of all temporal logic (sub)formulae obtained from the business model, i.e. $p_1 \wedge ... \wedge p_n$;

2) create the system's properties formula (formulae) Q expressed in temporal logic, which will be verified to meet the specifications of the previously obtained specification; such formulae can be entered manually, and it is also planned acquisition of such formulae from the model expressed in the language BPQL;

3) an automatic inference using semantic tableaux method (Semantic Tableaux Temporal Prover) for the complex formula $p_1 \wedge ... \wedge p_n \Rightarrow Q$.

Steps from 1 to 3, in whole or chosen, may be processed many times, e.g. whenever the specification of the BPMN model is changed (step 1), or there is a need to study further properties of the model (steps 2 and 3).

(A similar system as for BPMN diagrams can be created for the UML activity diagrams, however, this type of diagram is not in the mainstream of the considerations in this work. Some details on this can be found in work [Klimek et al. 2010].)

4. Generation of formulae

Obtaining formulae from BPMN or UML model, which is fundamental to the process of inference, seems possible due to the specification of these models. BPMN model is like a logical network with well-defined control flows while expressive power of UML activity diagram is very much similar. However, it will be considered below BPMN diagrams, and more precisely, their design patterns.

As already mentioned, further considerations are limited to *Propositional linear time temporal logic* PLTL having only two basic temporal operators $\Box$ and $\Diamond$. Temporal logics are usually interpreted (semantics) over Kripke structures (e.g. [Emerson 1990], which are graph structure having reachable states. Each state *S* is labeled with atomic propositions *AP* which are satisfied in this state. Labeling *L* is a function $L{:}S \rightarrow 2^{AP}$. These atomic propositions denote properties of the individual state. Analyzing BPMN graph, activities (tasks) of business model will be interpreted as an atomic activity *AA*. Atomic activities *AA* will be treated as atomic propositions *AP*. An activity is stisfied (fulfilled) when this activity was carried out and completed.

The use of workflow patterns while modeling in BPMN ensures obtaining better business models. Our considerations focus on patterns of BPMN models that belong to Basic Control Flows group. Let us consider five patterns.

Pattern Sequence Flow:

$$\Box(A \Rightarrow \Diamond B). \tag{2}$$

Pattern Parallel Split:

$$\Box(A \Rightarrow (\Diamond B \wedge \Diamond C)). \tag{3}$$

Pattern Synchronization:

$$\Box(A \Rightarrow \Diamond(B \wedge C)). \tag{4}$$

Pattern Exclusive Choice:

$$\Box(A \Rightarrow (\Diamond B \wedge \sim\Diamond C) \vee (\sim\Diamond B \wedge \Diamond C)), \tag{5}$$

$$\Box\sim(B \wedge C). \tag{6}$$

Pattern Simple Merge:

$$\Box((\mathit{true} \vee \sim\mathit{true}) \Rightarrow (\Diamond A \wedge \sim\Diamond B) \vee (\sim\Diamond A \wedge \Diamond B)), \tag{7}$$

$$\Box((A \vee B) \Rightarrow \Diamond C), \tag{8}$$

$$\Box\sim(A \wedge B). \tag{9}$$

Although some formulae, e.g. 2, 3, 4, etc., in fact belong to liveness properties, they must be preceded by □ operator. This follows from the fact that it is necessary to guarantee satisfiability of a formula for all possible executions in linear time domain, i.e. PLTL temporal logic.

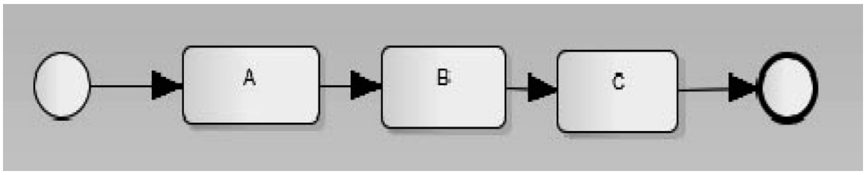

Figure 3. Sequence of two patterns sequence flow

Let us consider an example which refers to the sequential occurrence of two patterns of sequence flow. This is shown in Figure 3 (and also in Figure 1). Thus, it is a situation where some activities are executed one after the other. This case is relatively simple but it will allow us to trace the method of reasoning using semantic tableaux and temporal logic when examining the correctness of the model. The example consists of two patterns. The first is expressed by formula

$$\Box(A \Rightarrow \Diamond B). \tag{10}$$

The second is expressed by a similar formula

$$\Box(B \Rightarrow \Diamond C). \tag{11}$$

Of course, these two formulae could be extracted directly from the presented model. Then the correctness of the model expressed by the formula is verified.

$$\Box(A \Rightarrow \Diamond C). \tag{12}$$

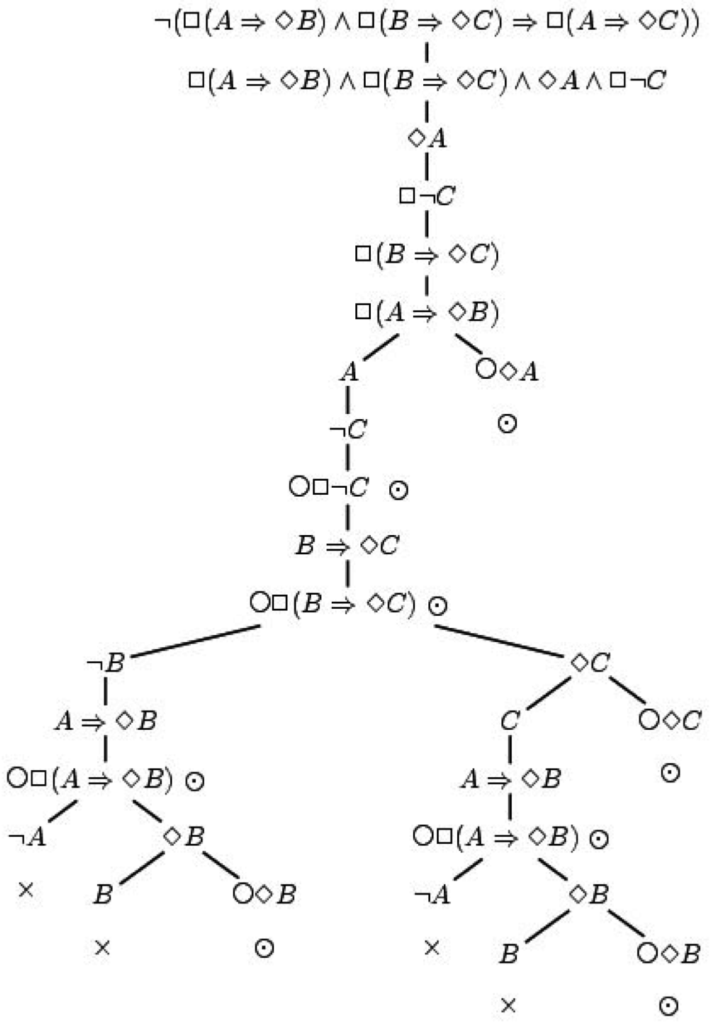

Figure 4. Decomposition tree of examined formula

It is interpreted in such a way that after the first activity, it is possible that the last activity of the diagram will be executed . In order to verify the model let us create the resulting formula which is the conjunction of all specification formulae and the implication of the desired property.

$$\Box(A\Rightarrow\Diamond B)\wedge\Box(B\Rightarrow\Diamond C)\Rightarrow\Box(A\Rightarrow\Diamond C). \tag{13}$$

Note, however, that at the root of the deduction tree a negation of the resulting formula is placed. The process for reasoning and decomposition is shown in Figure 4. Decomposition results in a development of the deduction tree and building up its branches. The purpose of the decomposition is to remove logical connectives to obtain elementary sentences. At the beginning of the procedure there is an attempt to get a conjunctive normal form. There is also used the law of duality for modal operators, i.e. $\sim\Diamond=\Box\sim$. Tableaux is completed which means that all branches of the tree are also completed, i.e. they either contain the contradictions (×) or node's expansion (means *expansion*, or *recursive equivalence*, *law* both for $\Box$ and $\Diamond$ operators) does not lead to new situations ($\odot$). Thus, the tree is completed and closed.

5. Conclusions, related works and future works

The paper presents new approach to business model verification which is based on temporal logic and semantic tableaux prover. The paper presents transformation algorithm from BPMN diagram (design patterns) to temporal logic formulae. Only BPMN design patterns [Russell et al. 2006; Van der Aalst et al. 2003] are considered since every business process might be modeled by combining common design patterns. By providing automatic transformation for these workflow patterns to temporal logic formulae it is possible to build automatically the logic specification for any given business process. The logic formulae are then processed using semantic tableaux methodology. As a result of this processing we might verify the related business model. (For modeling systems with UML the only constraint is to use activity diagrams to describe use cases.)

The example of the simple business process is given as BPMN diagram. The diagrams are transformed to the temporal logic formulae which are processed using semantic tableaux which might also be visualized. The advantage of the methodology is providing innovative concept of process verification which might be done for any given business model created with different notations: BPMN and UML. However, further research which includes providing transformation for the remaining workflow patterns. The advantages of the methodology are:

– automation of the process of inference,
– generating formulae from an existing business model,
– identifying locations in the inference tree, which contain errors of the business process,
– reducing software costs by allowing verification in the modelling phase.

Let us discuss some related works. The work of Eshuis and Wieringa [2003] addresses the issues of workflows, however, workflows are specified in UML activity diagrams and the goal is to translate diagrams into a format that allows states' exploration (model checking [Clarke et al. 1999]). It is worth noting that most of the works related to workflow verification applies only to statets' exploration algorithms. Another important direction of research is verifying the business processes using Petri nets [Van der Aalst 1998, 2002]. Different direction of the verification is to rely on the account of π-calculus [Ma et al. 2008], which was designed for business processes and BPEL environment. All of the research themes mentioned above are different from the approach presented in this work.

Future research should include several issues. Temporal logic formulae for all patterns of BPMN models should be developed. Next step is to develop rules for the submission and composition of patterns along with their formulae for larger blocks, i.e., not only atomic activities but also sub-processes. Although only the linear version of temporal logic was considered here, it is worthwhile to incorporate a computational tree logic which is related to branching time. It must be accompanied by production software implementation of the reasoning engine based on the method of semantic tableaux for temporal logic.[2]

References

Booch G., Rumbaugh J., Jacobson R. (1998), The Unified Modeling Language User Guide, Addison-Wesley, Reading, MA.

Business Process Modeling Notation Specification. Version 1.2 (2009), January, OMG Document dtc/2009-01-03.

Clarke E.M. Jr., Grumberg O., Peled D.A. (1999), *Model Checking*, MIT Press, Cambridge, MA.

Clarke E.M., Wing J.M. et al. (1996), Formal methods: State of the art and future directions, *ACM Computing Surveys*, Vol. 28, No. 4, pp. 626-643.

Emerson E.A. (1990), Temporal and modal logic, [in:] *Handbook of Theoretical Computer Science*, Vol. B: *Formal Models and Semantics*, Elsevier, MIT Press, Amsterdam–Cambridge, MA, pp. 995-1072.

Eshuis R., Wieringa R. (2003), Tool support for verifying UML activity diagrams, *IEEE Transactions on Software Engineering*, Vol. 30, No. 7, pp. 437-447.

Gamma E., Helm R., Johnosn R., Vlissides J. (1995), *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading, MA.

Handbook of Tableau Methods, Eds. M. D'Agostino, D.M. Gabbay, R. Hähnle, J. Posegga, Kluwer Academic Publishers, Dordrecht 1999.

Klimek R. (1999), *Wprowadzenie do logiki temporalnej*, Wydawnictwo Akademii Górniczo-Hutniczej, Kraków.

Klimek R., Skrzyński P., Turek M. (2010), Automatyczna weryfikacja modelu na etapie analizy wymagań, [in:] *Inżynieria oprogramowania w procesach integracji systemów informatycznych*, Eds. J. Górski, C. Orłowski, Pomorskie Wydawnictwo Naukowo-Techniczne, Gdańsk, pp. 209-216.

Ma S., Zhang L., He J. (2008), Towards formalization and verification of unified business process model based on pi calculus, [in:] *Proceedings of ACIS International Conference on Software Engineering Research, Management and Applications*, 2008, 1, IEEE Computer Society, pp. 93-101.

Riehle D., Zullighoven H. (1996), Understanding and using patterns in software development, *Theory and Practice of Object Systems*, Vol. 2, No. 1, pp. 3-13.

Russell N., ter Hofstede A.H.M., van der Aalst W.M.P., Mulyar N. (2006), *Workflow Control-Flow Patterns: A Revised View*, BPM Center Report BPM-06-22, BPMcenter.org.

Van Benthem J. (1995), Temporal logic, [in:] *Handbook of Logic in Artificial Intelligence and Logic Programming*, Vol. 4, Eds. Dov M. Gabbay, C.J. Hogger, J.A. Robinson, Clarendon Press, Oxford, pp. 241-350.

[2] Prototype of such systems has lately been developed.

Van der Aalst W.M.P. (1998), The application of Petri nets to workflow management, *The Journal of Circuits, Systems and Computers*, Vol. 8, No. 1, pp. 21-66.

Van der Aalst W.M.P. (2002), Making work flow: On the application of Petri nets to business process management, *Lecture Notes in Computer Science*, Vol. 2360: *Application and Theory of Petri Nets*, pp. 1-12, Springer-Verlag.

Van der Aalst W.M.P., ter Hofstede A.H.M., Kiepusewski B., Barros A.P. (2003), Workflow patterns, *Distributed and Parallel Databases*, Vol. 4, No. 1, pp. 5-51.

Wolper P. (1985), The tableau method for temporal logic: An overview, *Logique et Analyse*, Vol. 28, pp. 119-136.

WERYFIKACJA MODELI BIZNESOWYCH METODĄ DEDUKCYJNĄ

Streszczenie: Praca przedstawia formalną weryfikację procesów biznesowych wyrażonych w notacji BPMN. Weryfikacja jest oparta na wnioskowaniu logicznym. Została wprowadzona automatyczna transformacja wzorców procesowych BPMN do formuł logiki temporalnej. Formuły te składają się na logiczną specyfikację całego systemu i następnie są przetwarzane z wykorzystaniem metody tablic semantycznych. W ogólności, takie podejście do wnioskowania logicznego ma wiele zalet w porównaniu do podejścia tradycyjnego, opartego na metodzie rezolucji. W pracy pokazano automatyczną transformację formuł logicznych dla pięciu podstawowych wzorców procesowych BPMN, a także przedstawiono na diagramie BPMN pewnien prosty model biznesowy. Odpowiednie formuły logiki temporalnej uzyskiwane są w sposób automatyczny, po czym proces wnioskowania logicznego z wykorzystaniem metody tablic semantycznych pozwala poddać weryfikacji sam model biznesowy.