DEVELOPMENT OF THE APPROACH TO CHECK THE CORRECTNESS OF WORKFLOWS

ALEXANDER AFANASYEV, NIKOLAY VOIT, MARIA UKHANOVA, IRINA IONOVA

Ulyanovsk State Technical University, 32, Severny Venetz str. Ulyanovsk, 432027, Russia

The paper deals with an interesting approach for checking business process workflows. Business processes workflows are presented as a diagram based on graphical languages such as eEPC, UML, BPMN, IDEF0, and etc. We offer the approach based on a temporal grammar, a timed automaton and an ontology. It allows narrowing the semantic gap between business process analysis and business process execution. We propose to check the structural and semantic errors. Semantic errors are checked by the ontological model. The proposed approach can detect 23 errors, and the results are provided in visual form. The approach is illustrated by an example.

1. Introduction

Workflow is a trace for executing a set of business process tasks, taking into account time constraints and data flows. It is necessary to identify and correct errors in the processes in order to avoid failures. Although errors can occur in cause-effect relationships among tasks, we focus on the workflow execution's semantic errors, especially on denotative and significative semantics. Denotative semantics determines the errors of antonymy, words' synonymy in the workflow's business events. Significative semantics reveals workflow structural errors on the basis of trace isomorphism and homomorphism. Ad-hoc is an add-on in the workflow and makes the process not so strict, thereby it violates the canonical rules of the process. Such a workflow execution can lead to a customer's satisfaction decrease, an employee overload increase, a brand image decrease, a profits decrease, and a significant management time expenditure. Thus, it is important for business to identify and correct semantic errors in workflows.

The workflow should be conceptually presented in the formal language for analysis and expertise before deployment into a real business environment. This view is also useful when transferring workflow tasks between designers, users, process engineers, managers and technical personnel. In addition, process models in the presentation can be tested by approaches that have a corresponding formal language to determine a workflow. Conceptual representations can be performed using Workflow Nets (WF-nets), Workflow Graphs, Object Coordination Nets (OCoNs), Adjacency Matrix, Unified Modeling Language (UML) diagrams, Evolution Workflow Approach and Propositional Logic. Today test algorithms exist for WF-nets, Workflow Graphs, UML diagrams, Propositional Logic and Adjacency Matrix representations. And popular algorithms are those that are based on WF-nets and Workflow Graphs. WF-nets are based on Petri nets, and many formal methods for analyzing Petri nets are used to obtain theoretical solutions for problems encountered in the design of WF networks. Although many complicated structures of process language that are useful in a business environment can be implemented via WF-nets, the Workflow Management Council (WfMC) uses only six basic structures of process language. WfMC has adopted this approach to keep the simulation very simple and clear.

For a business event, a subset of workflow tasks is performed in accordance with the object data (customer data, environment data, business process data, and business domain data), for example, such as ordering. This subset of tasks, together with the workflow used to execute the business process, is called an instance. Until now, most workflow management systems (WfMSs) provide only modeling tools for testing workflow models via a trial and error method [1]. These modeling tools can be used to perform a subset of workflow instances to check for structural conflicts that may occur in the respective scenarios. However, workflows can have many instances, and the verification task becomes difficult for all instances.

Check for structural and semantic errors in workflows is a computational task, so different formal approaches and languages can be used for this. However, the approach taken for verification should support the language of the workflow description. Because of the computational complexity of a task (polynomial, exponential), only a few approaches successfully cope with the verification of workflows, taking into account constraints, including time constraints, for all types of workflow graphs.

The paper has the following structure. The list of standard problems with workflows is given in Introduction. The Related works paragraph has an overview

of works on this topic. In Temporal grammar, Timed automaton, Ontology and List of errors, we describe the approach. In Elaborate example, the Implementation presents our proposed approach. Results and the further directions of researches are in Conclusion.

2. Related works

We have studied many research works considered with the workflows' specification, verification and translation. Some of them focus on formal semantics and workflows verification methods using Petri nets, process algebra, and abstract state machine [5], [6]. Decker and Weske offer a formalism based on Petri Nets to define such properties as reliability and promptness, and a method for testing these two properties. However, they only describe the synchronous relationship and do not have any research comparisons for high-level interaction modeling languages as BPMN. Lohmann and Wolff offer the analysis using existing templates and monitoring them using compatible templates. In [3], the authors draw attention to the translation of BPMN into the process algebra for analyzing choreographies using the help model and checking equivalence. The Woflan tool was developed by H.W.M. Verbeek and W.M.P. Van der Aalst for checking structural conflict errors in WF-nets. The Woflan tool can also be used to test inheritance.

The main limit of the methods considered is that they do not work in different types of diagrams at the same time; it means that the input diagrams cannot be analyzed in some cases.

3. Temporal grammar

Temporal grammar (RVT-grammar) is defined as the tuple

$$G = (V, \Sigma, \tilde{\Sigma}, C, E, R, \tau, r_0). \quad (1)$$

where $V = \{v_e,\ e = \overline{1.L}\}$ is an additional alphabet for the operation onto a memory; $\Sigma = \{(a_l, t_l), l = \overline{1.T}\}$ is an alphabet (words) of events; $\tilde{\Sigma} = \{(\tilde{a}_n, \tilde{t}_n),\ n = \overline{1.\tilde{T}}\}$ is a quasi-term alphabet, extending Σ; $C = \{c_i, c_i = c_i + t_{l-1}, i \in N\}$ is a set of a time identifier, and a beginning $c_i = 0$; E is a set of the temporal relations as $\{c_i \sim t_l\}$, where c is a variable (a time identifier), $\sim \in \{=, <, \leq, >, \geq\}$; $R = \{r_i,\ i = \overline{0.I}\}$ is a rule of this grammar G (a set of production rule's complexes), where this complex r_i has a subset P_{ij} of the production rule $r_i = \{P_{ij},\ j = \overline{1.J}\}$; $\tau = \{t_l \in [0; +\infty], l = \overline{1.T}\}$ is a set of timestamps, where $c_i \in$

$\tau \times \sim \times \tau$; $r_0 \in R$ is an axiom of this grammar (a name of the first production rule), $r_k \in R$ is the last production rule. The production rule $P_{ij} \in r_i$ has a view as

$$(a_l, t_l) \xrightarrow{\{W_\gamma(\nu_1, \dots, \nu_n)|E\}} r_m. \quad (2)$$

where $W_\gamma(\nu_1, \dots, \nu_n)$ is *n*-relation, that defines a type of an operation over memory, depending on $\gamma = \{0,1,2,3\}$ (0 – operation is not performed, 1 – write, 2 – read, 3 – compare); (a_l, t_l) is a word as a pair of an event and a timestamp; $r_m \in R$ is a name of a target production rule. The language $L(G)$ of this grammar has words as (a_l, t_l) and presents a trace $\sigma = \{a_0, 0\} \to \{a_l, t_l\} \to \{a_k, t_T\}$.

4. Timed automaton

The timed automaton *TimedAutomaton* is represented by following components:

$$TimedAutomaton = (V, \sum, C, E, \delta, S_0, S, S_k), \quad (3)$$

where $V = \{\nu_e, e = \overline{1.H}\}$ is an auxiliary alphabet (the alphabet of operations over internal memory); $\sum = \{(a_l, t_l), l = \overline{1.L}\}$ is a terminal alphabet of a language; $C = \{c_i, c_i = c_i + t_{l-1}, i \in N\}$ is a finite set of clock identifiers, and a beginning $c_i = 0$; $E -$ is a set of time expressions C (clock limitation and clock reset), is limited by the following expressions: onwards $\{c_i \sim t_l\}$, and c_i is a variable, and t_l is a constant, $\sim \in \{=, <, \leq, >, \geq\}$; $S = \{S_i, i = \overline{0.I}\}$ is a set of states; $S_0 \in S$ is a beginning state; $S_k \in S$ is a ending state; the state transition function of automaton $\delta: S_i \times (a_l, t_l) \xrightarrow{\{W_\gamma(\nu_1, \dots, \nu_n)|E\}} S_m$ is the ratio of transitions, where $W_\gamma(\nu_1, \dots, \nu_n)$ is a *n*-th relation, which determines the type of operation over the internal memory depending on $\gamma \in \{0,1,2,3\}$ (respectively, 0 – operation is not performed, 1 – record, 2 – read, 3 – compare); $\nu_1, \dots, \nu_n \in V$; $r_i \in R$ is the name of the complex of a production rule's source; $S_m \in S$ is the name of the state of a production rule's successor.

5. Ontology

The ontology is presented as follows:

$$O = (Class, Property, Relation, Axiom). \quad (4)$$

where $Class$ is a set of concepts (classes) defined for a particular subject domain; $Property$ is a set of concept properties; $Relation$ is a set of semantic links

defined among concepts in $Class$. A set of relation types is the following: one to one, one to many and many to many. A set of basic relations is presented by: synonymy, a kind of something, part of something (*f*), instance of something, property of something (*property of*); $Axiom$ is a set of axioms. An axiom is a real fact or a rule that determines the cause-effect relationship.

6. List of errors

We can detect the following errors: 1. The cyclic link; 2. Mutually exclusive links; 3. Multiple links; 4. Remote context error; 5. Control transfer failure; 6. Input multiplicity error; 7. Output multiplicity error; 8. Invalid link; 9. Link error; 10. Access level error; 11. Message transmission error; 12. Control Transfer Error; 13. A quantitative error of diagram's elements; 14. Excluding links of a wrong type; 15. A call directed to the life line; 16. Dead link; 17. Dependency multiplicity violation; 18. Mutually exclusive links; 19. A synchronous call until a response; 20. Great synonymy; 21. Objects' antonymy; 22. Conversion of relationships; 23. Inconsistency of objects.

7. Elaborate example

Business processes as workflows are presented as a diagram based on graphical languages such as eEPC, UML, BPMN, IDEF0 using software of IBM, Whitestien Technologies, ARIS, OMG, etc.

The temporal property of this design process (workflow) is very important for designing and manufacturing, especially if we can manage time limits of production. The ARIS eEPC methodology gives all temporal properties of a design process as against UML, IDEF0, BPMN. Events and Functions are the main object in this methodology. Therefore, Events are represented by moments, and Functions are represented by decision-making processes. Events have a Frequency folder with attributes as the frequency of an event. Functions have a Simulation folder with attributes as a time period for making decision. The logical operations as AND, OR, XOR are control functions of workflows in this methodology.

Let's take a look at Figure 1 where it is described a sample of an approval process of design documentations. It usually takes three days to approve a design documentation for an assembly. This process consists of the Begin, Processing, Checking, Improve, Approval, End phases which are described on the basis of eEPC methodology.

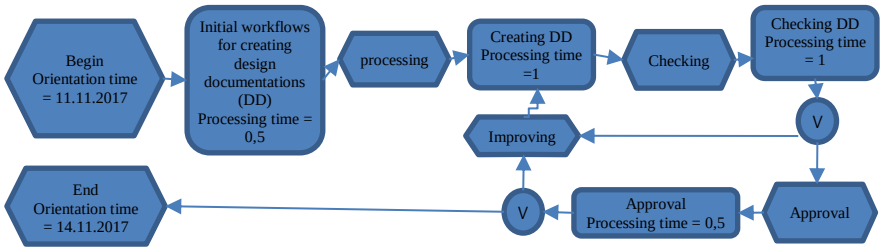

Figure 1. The approval of a design documentation based on eEPC methodology.

The Processing time attribute is presented in the above-mentioned folders (the sum of them is 3). The Orientation time attribute is presented in the Begin and End folders. We have developed a mathematical tool in order to analyze the temporal property of workflows by the temporal grammar. This tool helps us to dynamically reorganize parallel workflows for automated systems' lifecycle in a large enterprise. The goal is to reduce the idle time in manufacturing and time spent on an assembly.

Let's write grammar for Figure 1. The one automated store *m* and the one tape 1^t are used by this grammar as the internal memory. The timestamp tm is recorded on a tape. In Table 1, we can see the written RVT-grammar for Figure 1.

Table 1. RVT-grammar for Figure 1.

Source production rule's complex	**Quasi-term**	**Target production rule's complex**	**Relation**
r_0	B	r_1	$W_1(1^{1m}), W_1(tm^{1t})$
r_1	W	r_2	$W_1(2^{1m})/W_3(c<=tm^{1t})$
r_2	C	r_3	$W_1(3^{1m})/W_3(c>tm^{1t})$
r_2	C	r_4	$W_3(c<=tm_{1t})$
r_3	I	r_4	$c=0, W_2(3^{1m})/W_3(c<=tm^{1t})$
r_4	A	r_3	$W_2(3^{1m})/W_3(c>tm^{1t})$
r_4	A	r_k	$W_2(2^{1m}), W_2(1^{1m})$
r_k	E	–	–

Let's write a timed automaton for RVT-grammar. The alphabet of an event process is a set of $\sum = \{\text{Begin}, \text{Processing}, \text{Checking}, \text{Improve}, \text{Approval}, \text{End}\}$. $S = \{B, W, C, I, A, E\}$. Let's define the state transition function of automaton δ that has been formulated in a section Timed automaton (Table 2).

Table 2. A matrix of the state transition function of automaton δ.

Constraint	B	W	C	I	A	E
B		$W_1(1^{1m})$, $W_1(tm^{1t})$				
W			$W_1(2^{1m})$/ $W_3(c<=tm^{1t})$			
C				$W_1(3^{1m})$/ $W_3(c>tm^{1t})$	$W_3(c<=tm^{1t})$	
I					c=0, $W_2(3^{1m})$/ $W_3(c<=tm^{1t})$	
A				$W_1(3^{1m})$/ $W_3(c>tm^{1t})$		$W_2(2^{1m})$, $W_2(1^{1m})$
E						

Let's depict an ontology for this timed automaton. We transform the automaton into an ontology, replacing the *States* with *Classes*, adding properties to the notions (*Property*). We get a graphical representation of the ontology with class properties (Figure 2).

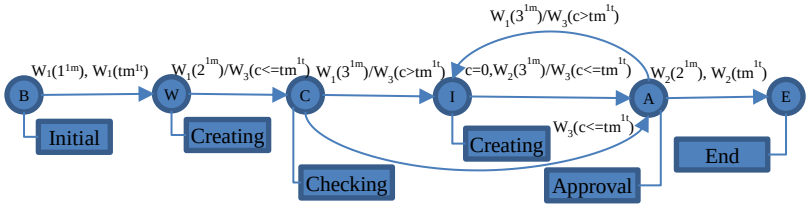

Figure 2. The ontology for Figure 1.

8. Discussion

In Figure 2, we see that *Class* W and I have the *Creating* property, so you can say that the properties of these classes are synonymous and you need to establish a synonymy relationship among these classes. Also, classes B and E are antonyms, so an antonymy relationship can be established between them. Thus, it is possible to structure an ontology and identify semantic errors, including isomorphism and homomorphism, according to List of errors.

Conclusion and future works

We have developed an approach in order to analyze errors according to the list of errors in business processes workflows. This research work is different from existing ones in the fact that it has checked not only structural errors but also semantic errors. The proposed temporal finite-state grammar has a linear characteristic of time analysis of a workflow, takes into account the process description language and can be applied to any diagram. A timed automaton

allows modeling a process in visual form. Analysis for denotative and significative errors in workflows is based on the ontological model. We developed a list of structural and semantic errors encountered in workflows.

Our future works will present examples of the approach application in industry, training, cyber-physical systems, in the development of automated systems.

Acknowledgments

The reported study was funded by RFBR according to the research project № 17-07-01417.

References

1. H. H. Bi, J. L. Zhao, *Information Technology and Management*, **5**, 293 (2004).
2. P. Poizat, G. Salaün, Checking the Realizability of BPMN 2.0 Choreographies, in *Proc. of* SAC'12 (2012)**.**
3. A. N. Afanasyev, N. N. Voit and S. Y. Kirillov, Development of RYT-grammar for analysis and control dynamic workflows, in *Proc. of International Conference on Computing Networking and Informatics (ICCNI)*, (Lagos, Nigeria, 2017). doi: 10.1109/ICCNI.2017.8123797, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8123797&isnumber=8123766.
4. A. Afanasyev, N. Voit, R. Gainullin, The analysis of diagrammatic models of workflows in design of the complex automated systems, in *Proc. of International conference on Fuzzy Logic and Intelligent Technologies in Nuclear Science* (FLINS2016), (France, Roubaix, 2016).
5. Y. Wang, Y. Fan, Using Temporal Logics for Modeling and Analysis of Workflows, in *Proc. of E-Commerce Technology for Dynamic E-Business, IEEE International Conference on*. doi: 10.1109/CEC-EAST.2004.72 (2004).
6. N. Saeedloei and G. Gupta, Timed definite clause ω-grammars, In *Proc. of/ Technical Communications of the International Conference on Logic Programming* (2010). URL: http://www.floc-conference.org/ICLP-home.html