Denotative and Significative Semantics Analysis and Control Methods in Diagrams

Nikolay Voit[1][0000-0002-4363-4420] and Semen Bochkov[0000-0003-1089-4119]

[1] Ulyanovsk State Technical University, Severny Venets str. 32, 432027 Ulyanovsk, Russia
n.voit@ustu.ru, bochkovsi@ido.ulstu.ru

Abstract. Authors have proposed algorithms for the analysis and control of the design workflows diagrammatic models' denotative and significative semantics. The denotative characteristics are the basis for the denotation description as a graphic language concept; therefore, a dictionary of such graphic words is formed in the work. These characteristics allow to determine the synonyms and antonyms of graphic words, due to which it is possible to eliminate errors in understanding the diagram operation (semantic errors). Significative semantics based on isomorphism and homomorphism of workflow traces analysis reveals structural errors.

Keywords: Workflows, Graphic Languages, Antonymy, Synonymy.

1 Introduction

During CAD systems design, diagrammatic models are actively used, presented in artifacts of visual graphic languages BPMN, UML, IDEF and others. This significantly increases the design process efficiency and the quality of the created systems due to the process participants interaction language unification, rigorous documentation of design and architectural, functional solutions and formal control of diagram correctness.

At the same time, an obligatory step in enterprise business processes modeling is automatic and/or automated verification of the obtained models. Defect-free completion analysis issues are relevant and actual, since the complexity of models is constantly increasing, and the verification tools built into the simulation environment are far from perfect.

The variety of diagrammatic graphic languages covers all possible system descriptions types; however, unsolved problems exist. Graphic design support tools do not use universal methods of parsing and are highly specialized and aimed at working with few graphic languages. The control tools development for new languages takes considerable time, because it actually requires a new analyzer creation from the scratch. Known methods of parsing have significant costs in time (exponential, polynomial characteristics) and memory.

2 Related Works

The Common Workflow Language (CWL) workflows descriptors are researched in [1], which allow analyzing data in various computing environments (Docker containers virtualization). The authors developed CWL-metrics, a utility tool for cwltool (a reference implementation of CWL), for collecting Docker container runtime metrics and workflow metadata for analyzing resource requirements. To demonstrate the use of this tool, the authors analyzed 7 workflows on 6 types of instances. Analysis results show that choosing the type of instance allows to reduce financial costs and execution time using the required amount of computing resources. However, in the CWL-metrics implementation proposed by the authors, there are no functions for collecting metric data from parallel tasks in workflows.

[2] is devoted to the dynamic access control approach for business processes development. Authors offer a context-oriented and trust-oriented work environment. The proposed approach focuses on inter-component relationships, where steps are performed online or offline to avoid performance bottlenecks. It should be noted that the presented context-oriented access structure is applicable only for solving problems related to business processes in service-oriented computing.

[3] described a new approach to the systematic support of engineers using model-driven system architectures for process design and plant automation. The authors investigated a new aspect the virtual intelligent objects design in enterprise data models, which represents the life cycle of an object. A methodology is described that enables users to define the life cycle for classes of objects depending on the context and goals of the projects. The authors performed workflows research and analysis to form a library of production processes for certain objects classes. However, the dynamic distributed workflows analysis and control methods are not considered.

In [4], an approach to the selection of services for modeling business processes is proposed. At the first stage, the function similarity method is used to select services from the service repository to create a set of candidate services that checks the description of functions to find suitable services, especially a service can publish one or more functions through several interfaces. At the second stage, a method based on the probabilistic model verification, which includes the composition of services and calculation of stochastic behavior in accordance with the workflow structures, is used to quantitatively verify process instances. Next, experiments are carried out to demonstrate the efficiency and effectiveness of the proposed method compared to traditional methods.

In [5], an improved two-stage approach of exact query based on the graph structure is proposed. At the filtering stage, a composite task index, which consists of a label, a connection attribute and a task attribute, is used to obtain candidate models, which can significantly reduce the number of process models that need to be tested at a specific time - the verification algorithm. At the verification stage, a new subgraph isomorphism test based on the task code is proposed to clarify the set of candidate models. The experiments are conducted on six synthetic model kits and two real model kits. However, the algorithm has polynomial computational complexity.

In [6], success and failure factors for the implementation of business process management technologies in organizations were investigated.

Workflow mining in highly changing areas is researched in [7], focusing on creating models of process instances (in the form of instance graphs) from simple event logs.

In the [8], according to the author, the most solid foundation in the theory of workflows at present is Petri nets. In the analysis, the author identifies the following errors: blocked tasks, deadlock - "freezing", active deadlock (endless loop), performing tasks after reaching the end point, the presence of chips in the network after the network is shut down, and others. He also claims that most modern workflow modeling languages (BPMN, EPCs, FileNet, etc.) are built on WF networks, a subclass of Petri nets. However, taking into account the very wide distribution of tools for working with BPMN- and similar diagrams, for applying the author's methods there is a need for initial translation of the original diagrams into the Petri net, which leads to unnecessary costs.

In [9], the authors extend the generalized LR algorithm to the case of "grammars with a left context", which supplement context-free grammars with special operators for referencing the left context of the current substring, as well as a join operator (as in conjunctive grammars) to combine syntactic conditions. All the usual components of the LR algorithm, such as a parsing table, shift and reduce actions, etc., are extended to handle context statements. The resulting algorithm is applicable to any grammar with a left context, but it has the same performance in cubic time for the worst case. Also in [10], variants of union and concatenation operations in formal languages are studied in which Boolean logic in definitions (that is, conjunction and disjunction) is replaced by operations in the two-element field GF (2) (conjunction and exclusive OR). It is argued that the computational complexity is the same as that of conventional grammars with concatenation and concatenation: in particular, a simple time analysis of O (n^3), which is worse than the linear performance presented in the project for the RVTI-analyzer.

In article [11], the author writes that the LL parsing algorithm allows arbitrary context-free grammars and achieves good performance, but cannot process EBNF-grammars. The main contribution of this article is a modification of the GLL algorithm, which can process grammars in a form closely related to EBNF (extended context-free grammar). As can be seen from the work, the performance increases, however, it still has a power-law growth with an increase in the elements of the analyzed chain.

A number of Russian scientists are involved in ontology engineering issues that are used as the basis of the proposed method: Gavrilova T.A., Vagin V.N., Gribova V.V., Zagorulko Yu.A., Kleschev A.S., Palchunov D. E., Smirnov S.V., Kureichik V.M., and others. The "Design Ontology" magazine is published, which contains sections "Applied Design Ontologies", "Ontology Engineering", and "General Design Issues: Ontological Aspects".

For example, in [12], when designing decision-making systems at a conceptual level, an ontological approach is used.

In [13], work is carried out with ordinary texts, and not with diagrammatic models. LSPL is used to describe lexical and syntactic patterns. The disadvantage is the lack of control after data extraction.

The work of science school led by prof. Yarushkina N.G. [14] is the closest to the set problem. The article presents the selection and justification of the notation of design diagrams for the description of technological processes using the example of a fragment of the model for the description of the technological process "Assembly of the door

frame" at the CJSC Aviastar-SP. A description of the approach to the transformation of UML – OWL and the search for a similar software project from the repository by the severity of common design patterns using ontological engineering methods is given. However, the work does not address the issues of semantic analysis of diagrammatic workflows based on the linguistic approach.

The [15] presents mathematical models of granularity and graduality. The author offers an improved method of fuzzy granulation. The practical application of fuzzy granulation in the tourism sector is considered, however, there is no connection with grammars in the work.

The problem of error neutralization and its solution is well reflected in classical works on compilers [16]. An error neutralization method for RV-grammars has also been proposed [17]. However, they did not solve the issues of neutralization for diagrammatic models of dynamic distributed workflows.

Translation of visual language models into another target language based on RV-grammars is solved in [18]. However, the task of translating several interconnected diagrammatic models of workflows presented in different languages into the target language is not considered.

In [19], the system of transformations of grammars SynGT is used, syntactic graph-schemes are investigated. However, automatic synthesis of grammars is not considered. The [20] uses grammars with generalized regular expressions and attributes in the form of semantics to represent contextual constraints and syntactic graph-schemes. The vocabulary, DAC rules and contextual conditions of the Yard language are given. The SynGT software system was developed, which includes a text editor, a graphic editor, equivalent syntactic transformations that form a grammar, and a test generator. Grammar is not synthesized, but the recognizer works with any DAC grammar. However, the visual languages used in the practice of designing the workflow do not belong to the DAC languages, but are more complex.

The [21] is devoted to the problem of finding frequent and similar fragments in work processes using graph analysis methods. The authors examine various representations that can be used to encode workflows before evaluating their similarity, taking into account the efficiency and effectiveness of the data mining algorithm. However, the issue of building a library or repository of such workflows for reuse is not addressed.

Thus, related works analysis above shows the lack of semantic errors analysis and control methods including syntactic (topological) and temporal ones in diagrammatic models of hybrid dynamic design workflows represented in the basis of graphic languages such as BPMN, UML, IDEF, eEPC, etc. The authors have developed a temporal automatic RVTI-grammar [24] as the basis of the mathematical apparatus for parsing and identifying semantic errors in diagrams.

3 Mathematical Apparatus

In CAD systems, hybrid dynamic workflows diagrammatic models denotative semantics control and analysis methods cooperate with synonyms and antonyms of temporal

words in graphical languages. The goal is to detect errors in diagrammatic models events and thereafter correct them.

The method differs from similar ones in functioning with hybrid dynamic diagrammatic design workflows models, temporal graphic words of denotative semantics. The method also has a linear law of analysis time complexity. The hybrid dynamic design workflows diagrammatic models control and analysis method in denotative semantics contains procedures for diagrammatic models and denotative semantics analysis.

Consider the procedure for analyzing a hybrid dynamic diagrammatic model (diagram) of design workflows. The input is a diagram, its model has the following view: G = (V,E,TV,TE), where:

- V is vertices set,
- E is edges set, $E \subset (V \times V)$,
- TV is vertices types set,
- TE is edges types set.

The output is correctness status of diagram with an error message if exists.

The diagrammatic model analysis procedure has the following steps.

Step 1. Define the initial vertices search function.

For each diagram type the initial vertices search function (Fstart: $V \rightarrow \{0,1\}$) must be defined. Start the traversing process for the diagram analysis from these vertices.

Step 2. Define an automaton to control the bypass process.

For the traversing process management define a finite state machine A of RVTI-grammar [24].

A = (S, T, S_0, C, Send, Ftrans), where

- S is states set,
- T is input chars (terms) set,
- S_0 is an initial state,
- C is transitions conditions set,
- FA is transitions functions set,
- Send is finite states set,
- Ftrans is a transition function: S x T x C -> S x FA.

Step 3. Launch the diagram traversing algorithm.

Initial terms set START_V is formed for the diagram. It is written in the stack STACK. Then the first element T is popped from the stack to the state machine input.

Step 4. The automaton searches for a rule in an ordered list of rules corresponding to the current state of the automaton, the input term and the transition condition, and also checks for the denotative errors signs, according to the sign of denotative and significative errors (the procedures for analyzing denotative semantics are given below).

4.1. If there is no rule, the execution ends, go to 4.5.

Otherwise, the automaton switches to the next state. When navigating, bound functions are executed.

4.2. Unanalyzed adjacent terms are added to the stack STACK.

4.3. If the next state belongs to the finite states set, go to 4.5.

4.4. For the current term, a list of the following terms is formed and fed to the state machine input, go to 4.5.
4.5. Completing the traversal.
Step 5. State checking. If the current automaton state belongs to the final states set and all the diagram terms are analyzed, then the diagram is considered correct. Otherwise, an error message is generated.

The procedure for analyzing denotative semantics (synonyms) in diagrams for errors is as follows:

1. Upload the synonymous groups list.
2. Extract from the diagrammatic model (diagram) all vertices names list.
3. For each term from the names list:
   a. Do the term lemmatization, or bring it to normal form.
   b. Find a term in existing local synonym groups.
   c. If the group is not found, create a new synonymous group and add it to the list.
   d. Add the term to the selected synonym group.
   e. Increase the synonymous group usage counter by 1.
4. Select all local synonym groups where the usage counter is greater than 1.
5. For each selected synonymic group, compare the values of all user parameters for each vertex.
6. If vertices are found that belong to the same synonymous group, but with different user parameters, output a message about non-matching parameters within the synonymous group.

The procedure for analyzing denotative semantics (antonyms) in diagrams for errors is as follows:

1. Upload the antonymous groups list.
2. Extract from the diagrammatic model (diagram) all vertices names list.
3. For each term from the names list:
   a. Do the term lemmatization, or bring it to normal form.
   b. Find a term in existing local antonym groups.
   c. If the group is not found, go to the next term.
   d. Add the term to the selected antonym group.
4. For each selected antonymic group, compare the values of all user parameters for each vertex.
5. If vertices are found that belong to the same antonymous group, but with the same user parameters, output a message about non-matching parameters within the synonymous group.

Consider the BPMN diagram analysis on the example given in the Fig. 1. Elements attributes values are presented as (Categories,), (Documentation,), (LoopType, None), (TaskType, None), (IsForCompensation, FALSE), (BoundaryType, Default), (Initial value, 20 degrees), (Period, 10 minutes), (Duration, 30 seconds) for «Increase», and as (Categories,), (Documentation,), (LoopType, None), (TaskType,

None), (IsForCompensation, FALSE), (BoundaryType, Default), (Initial value, 30 degrees), (Period, 15 minutes), (Duration, 120 seconds) for «Add».

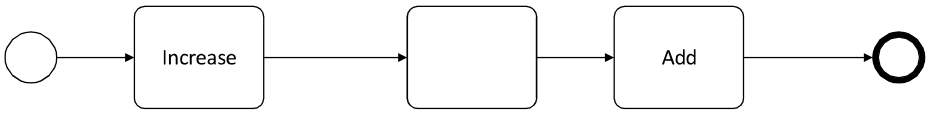

Fig. 1. Example of the BPMN diagram with synonym group

In order to start the analysis, it is necessary to have the synonym groups dictionary or ontology as [25], the one's example is shown in the Listing 1.

```
[ [ "Start", "Begin", "Initial" ], [ "Finish", "End" ], [ "Heat", "Warm up", "Heat up" ], [ "Add", "Grow up", "Increase", "Upgrade" ],...]
```

Listing 1. Synonym groups dictionary example in JSON format

BPMN diagram analysis result is shown in the Fig. 2.

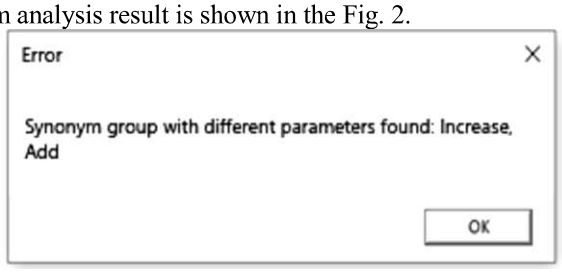

Fig. 2. BPMN diagram analysis example with synonym group

In this case, the "Increase" and "Add" elements have a similar name, but different values for the "Initial Value", "Period", and "Duration" parameters.

The proposed analysis method is implemented on the .Net Framework 4.5 for analyzing BPMN diagrams made in Microsoft Visio 2017.

Hybrid dynamic workflows diagrammatic models significative semantics control and analysis methods works with isomorphism, homomorphism of temporal traces (tracks) of diagrammatic models of a graphic language in order to identify structural errors in diagrammatic models for subsequent transformations of these traces. The method differs from its analogues in that it works with hybrid dynamic design workflows diagrammatic models, temporal graphic words of significative semantics and has a linear law of time complexity of analysis.

Hybrid dynamic workflows diagrammatic models significative semantics control and analysis methods contain a procedures for analyzing diagrammatic models and significative semantics.

The diagrammatic model analysis procedure has the following view:

Step 1. Define the initial vertices search function.

For each diagram type the initial vertices search function (Fstart: V → {0,1}) must be defined. Start the traversing process for the diagram analysis from these vertices.

Step 2. Define an automaton to control the bypass process.

For the traversing process management define a finite state machine A of RVTI-grammar [24].

A = (S, T, S_0, C, Send, Ftrans), where

- S is states set,
- T is input chars (terms) set,
- S_0 is an initial state,
- C is transitions conditions set,
- FA is transitions functions set,
- Send is finite states set,
- Ftrans is a transition function: S x T x C -> S x FA

Step 3. Launch the diagram traversing algorithm.
Initial terms set START_V is formed for the diagram. It is written in the stack STACK. Then the first element T is popped from the stack to the state machine input.
Step 4. The automaton searches for a rule in an ordered list of rules corresponding to the current state of the automaton, the input term and the transition condition, and also checks for the significative errors signs, according to the sign of denotative and significative errors (the procedures for analyzing significative semantics are given below).
4.1. If there is no rule, the execution ends, go to step 5.
Otherwise, the automaton switches to the next state. When navigating, bound functions are executed.
4.2 Unanalyzed adjacent terms are added to the stack STACK.
4.3 If the next state belongs to the finite states set, go to step 5.
4.4 For the current term, a list of the following terms is formed and fed to the state machine input, go to step 5.
4.5 Completing the traversal.
Step 5. State checking. If the current automaton state belongs to the final states set and all the diagram terms are analyzed, then the diagram is considered correct. Otherwise, an error message is generated.

The procedure for analyzing significative semantics (relations convertibility) in diagrams for errors is as follows:

1. Upload the antonymous groups list.
2. Extract from the diagrammatic model (diagram) all vertices names list.
3. For each term from the names list:
   a. Do the term lemmatization, or bring it to normal form.
   b. Find a term in existing local antonym groups.
   c. If the group is not found, go to the next term.
   d. Add the term to the selected antonym group.
4. For each selected antonymic group, check for the convertible relations between elements.
5. If relations are not found, output the absence of convertibitlity relations.

The procedure for analyzing the significative semantics (objects inconsistency) for errors in diagrams is as follows:

1. Split charts into unrelated ones.
2. For each diagram, define a diagram description.
3. For each diagram:
    a. Select all vertices included in the diagram.
    b. Check each vertex for a suitable nested diagram from the available diagrams.
    c. If a suitable nested diagram is found, output a message about the lack of connection between the selected vertex and the nested diagram.

Consider the BPMN diagram analysis on the example on the Fig. 3. Element attributes values are presented as (Categories, Documentation), (LoopType, None), (TaskType, None), (IsForCompensation, FALSE), (BoundaryType, Default), (Initial value, 200 degrees), (Period, 100 minutes), (Duration, 300 seconds) for «Increase», and (Categories, Documentation), (LoopType, None), (TaskType, None), (IsForCompensation, FALSE), (BoundaryType, Default), (Initial value, 200 degrees), (Period, 100 minutes), (Duration, 300 seconds) for «Decrease».

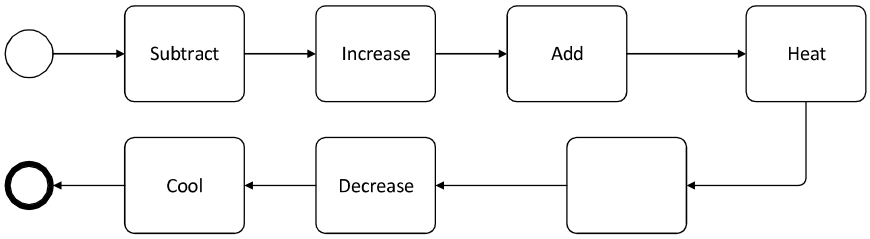

Fig. 3. Example of BPMN diagram with antonym groups

In order to start the analysis, it is necessary to have the antonym groups dictionary or ontology as [25], the one's example is shown in the Listing 2.

```
[ [ "Add", "Subtract" ], [ "Increase", "Decrease" ], [ "Begin",
  "End" ], [ "Heat", "Cool" ], [ "Simplify", "Complicate" ], [ "Find",
  "Lose" ], [ "Switch on", "Switch off" ]...]
```

Listing 2. Antonym groups dictionary example in JSON format

BPMN diagram analysis result is shown in the Fig. 4.

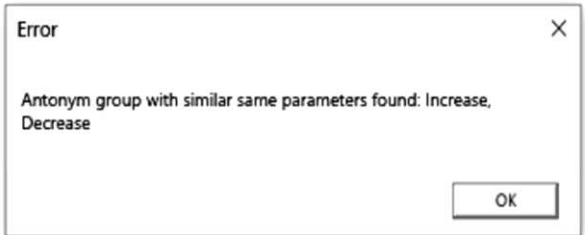

Fig. 4. BPMN diagram analysis example with antonym group

As can be seen, in this case «Increase» and «Decrease» elements have controversial names but the same values of all attributes.

The proposed analysis method is implemented on the .Net Framework 4.5 for analyzing BPMN diagrams made in Microsoft Visio 2017 [22, 23].

The semantic errors of hybrid dynamic diagrammatic workflow models are as follows.

Synonyms mismatching (denotative error).

Temporal words $(\tilde{a}_l, t_i)$ and $(\tilde{a}_k, t_i)$ of visual language are synonyms if and only if $\tilde{a}_l \neq \tilde{a}_k, \tilde{a}_l \equiv \tilde{a}_k, t_i < t_j$ and the synonyms of words are denoted as $(\tilde{a}_l, t_j) \equiv (\tilde{a}_k, t_j)$. The identical equality of the words determines the similarity of the structure and meanings of the denotation signs. A mistake is a situation when the word denotations names in two temporal traces of a graphic language are similar, but the meanings of other features are very different. In practice, this situation is presented as follows, when analyzing the diagrammatic model of the visual language, the structural similarity of words and names of denotations is revealed, but the meanings of the remaining signs of words denotations are different. To represent options for the composition of products under various conditions: in versions, replaceability and interchangeability, in this situation, the interchangeability implementation of such words in the diagrammatic model of the visual language is an error of synonyms mismatching.

Antonyms mismatching (denotative error).

Temporal words of visual language $(\tilde{a}_l, t_i)$ and $(\tilde{a}_k, t_i)$ are antonyms if and only if $\tilde{a}_l = \neg\tilde{a}_k, t_i < t_j$ and the antonyms of words are denoted as $(\tilde{a}_l, t_l) \equiv (\neg\tilde{a}_k, t_k)$. The identical opposition of two words determines the similarity of the structure and the opposition (inversion) of the meanings of the denotation signs. Typically, the words "Start" and "End" in a diagram are antonyms of the graphic language. An error is a design situation when the names of words denotations in two temporal traces of a graphic language are opposite (inverse), but the meanings of other features are very similar. In practice, this situation is presented as follows, when analyzing the diagrammatic model of the visual language, the structural similarity of words and the inversion of the denotations names are revealed, but the meanings of the remaining signs of words denotations are similar. In this situation, considering the opposite meaning of these words in the diagrammatic model of the visual language is a mistake of antonyms mismatching.

Relations convertibility consists in linking the antonyms of the visual languages diagrammatic models that describe the same project situation, but with different roles (different organizational levels). The convertibility error of relations is significative, that is, structural (constructional), and is defined as the absence of these relations between the antonyms of diagrammatic models that describe the same project situation, but with different roles.

The objects inconsistency is a significative error that means the absence of a relationship between dependent temporal words. Analysis on nested diagrams allows to identify errors in the absence of a link between the element and the nested diagrammatic model.

List of syntactic (topological, structural) and semantic errors in hybrid dynamic design workflows in various bases of graphic languages is given below.

BPMN graphical language:

1. Synonyms group with different parameters.
2. Antonyms group with the same parameters.
3. Lack of conversion link.
4. Lack of nested connection with the diagram.

EPC graphical language:

1. No start character.
2. Too many outgoing connections.
3. Too many incoming connections.
4. Incomplete sequence.
5. No ending.
6. Deadlock.
7. Unparsed element.
8. The next figure was expected.

IDEF3 graphical language:

1. There are more than 1 initial characters.
2. No start character.
3. Too many outgoing connections.
4. Too many incoming connections.
5. Incomplete sequence.
6. No ending.
7. Deadlock.
8. Unparsed element.
9. The next figure was expected.

IDEF5 graphical language:

1. Incomplete sequence.
2. No ending.
3. Unparsed element.
4. The next figure was expected.
5. No start character.
6. Unknown symbol.
7. Many exits.
8. No exits.
9. The peak was expected.
10. Incorrect communication type.
11. Incorrect vertex type.

4 Conclusion

The proposed RVTI-grammar and methods, software and information tools are the contribution and development of the following science areas:

1. Theory and practice of design, development and maintenance of automated systems.
2. The formal languages and grammars theory.
3. Theory and practice of processing visual and graphic languages.

New analysis and control methods in denotative and significant semantics and semantics of diagrammatic design workflows models have been developed. They provide the quality improvement of such diagrams and expand the theoretical foundations of the business process management theory.

Scientifically, the author's RVTI-grammar and analysis methods provide a linear time law for the analysis of the diagram because of the automaton approach and analytical proof of its usage.

In practice, the author's RVTI-grammar and methods improve the quality of diagrams, including workflows, by identifying complex semantic errors at the conceptual stage of technical systems development, which provides an economic positive effect.

In future works, the outlook is to interpretation methods of these diagrams in various bases of graphic languages, including the Unified Model Language (UML), Business Process Model Notation (BPMN), etc.

Acknowledgments. This research was funded by Russian Foundation for Basic Research and the government of the region of the Russian Federation, grant № 18-47-730032.

References

1. Ohta, T., Tanjo, T., Ogasawara, O.: Accumulating computational resource usage of genomic data analysis workflow to optimize cloud computing instance selection. GigaScience, 8 (4), 1-11 (2019).
2. Bhattasali, T., Chaki, N., Chaki, R., Saeed, K.: Context and trust aware workflow-oriented access framework. In: Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE-2016, http://ksiresearchorg.ipage.com/seke/seke16paper/seke16paper_179.pdf, last accessed 2020/06/01.
3. Bigvand, G., Fay, A.: A workflow support system for the process and automation engineering of production plants. In: 2017 IEEE International Conference on Industrial Technology (ICIT), pp. 1118-1123 (2017).
4. Gao, H., Chu, D., Duan, Y., Yin, Y.: Probabilistic Model Checking-Based Service Selection Method for Business Process Modeling. International Journal of Software Engineering and Knowledge Engineering, 6 (27), 897–923 (2017).
5. Huang, H., Peng, R., Feng, Z.: Efficient and Exact Query of Large Process Model Repositories in Cloud Workflow Systems. IEEE Transactions on Services Computing, 11 (5), 821-832 (2018).
6. Reijers, H.A., Vanderfeesten, I., van der Aalst, W.M.P.: The effectiveness of workflow management systems: A longitudinal study. International Journal of Information Management, 36 (1), 126-141 (2016).
7. Diamantini, C., Genga, L., Potena, D., van der Aalst, W.M.P.: Building instance graphs for highly variable processes. Expert Systems with Applications, 59, 101-118 (2016).

8. van der Aalst, W.M.P.: Everything You Always Wanted to Know About Petri Nets, but Were Afraid to Ask. In: Hildebrandt T., van Dongen B., Röglinger M., Mendling J. (eds) International Conference on Business Process Management, pp. 3-9. Springer, Cham (2019).
9. Barash, M., Okhotin, A.: Generalized LR Parsing Algorithm for Grammars with One-Sided Contexts. Theory of Computing Systems, 61, 581-605 (2016).
10. Bakinova, E., Basharin, A., Batmanov, I., Lyubort, K., Okhotin, A., Sazhneva, E.: Formal Languages over GF(2). In: Klein S., Martín-Vide C., Shapira D. (eds) Language and Automata Theory and Applications. LATA 2018. Lecture Notes in Computer Science, vol. 10792, pp. 68-79. Springer, Cham (2018).
11. Gorokhov, A., Grigorev, S.: Extended Context-Free Grammars Parsing with Generalized LL. In: Itsykson V., Scedrov A., Zakharov V. (eds) Tools and Methods of Program Analysis. TMPA 2017. Communications in Computer and Information Science, vol 779., pp. 24-37. Springer, Cham (2018).
12. Sorokin, A. B.: Conceptual Design of Intelligent Systems of Decision Making Support. Ontology of Designing, 3 (25), 247-269 (2017).
13. Galieva, Yu. A.: Automated construction of databases from natural language data sources. In: Proceedings of Russian and international scientific conference of young scientists «Mathematics and Interdisciplinary researches», pp. 39-45. Perm State University, Perm (2017).
14. Yarushkina, N.G., Afanasyeva, T.V., Negoda, V. N., Samohvalov, M. K., Namestnikov, A. M., Guskov, G. Yu., Romanov, A. A.: Integration of project diagrams and ontologies in the aircraft manufacturing enterprise capacity balancing task. Automation of Control Processes 4 (50), 85-93 (2017).
15. Lisi, F. A., Mencar, C.: A System for Fuzzy Granulation of OWL Ontologies. In: Petrosino A., Loia V., Pedrycz W. (eds) Fuzzy Logic and Soft Computing Applications. WILF 2016. Lecture Notes in Computer Science, vol. 10147, pp. 126-135. Springer, Cham (2016).
16. Aho, A., Sethi, R., Ullman, J., Lam, M.: Compilers: Principles, Techniques, and Tools. 2nd edition. Addison-Wesley (2006).
17. Sharov, O.G., Afanas'ev, A.N.: Neutralization of syntax errors in graphic languages. Programming and Computer Software 1 (34), 61-66 (2008).
18. Sharov, O. G., Afanas'ev, A. N.: Methods and tools for translation of graphical diagrams. Programming and Computer Software 3 (37), 171–179 (2011).
19. Fedorchenko, L. N.: Generation of Tests in The SynGT System. BSU bulletin. Mathematics, Informatics 2, 33-39 (2017).
20. Fedorchenko, L. N.: Construction of The Yard Language Recognizer Using Syntax Graph-Scheme. BSU bulletin. Mathematics, Informatics 2, 33-39 (2017).
21. Harmassi, M., Grigori, D., Belhajjame, K.: Mining workflow repositories for improving fragments reuse. In: Cardoso J., Guerra F., Houben GJ., Pinto A., Velegrakis Y. (eds) Semantic Keyword-based Search on Structured Data Sources. IKC 2015. Lecture Notes in Computer Science, vol 9398, pp. 76-87. Springer, Cham (2016).
22. Voit, N.N., Kirillov, S.Y., Bochkov, S.I.: Converting Diagram to a Timeline Ontology. In Proceedings of the 6th International Conference on Computer and Technology Applications (ICCTA'20), pp. 80-86. ACM, New York, NY, USA (2020).
23. Voit, N.N., Ukhanova, M.E., Kirillov, S., Bochkov, S.I.: Method to Create the Library of Workflows. In: P. Sosnin, V. Maklaev, E. Sosnina (eds.): Proceedings of the IS-2019 Conference (IS'2019), pp. 97-107. UlSTU, Ulyanovsk, Russia (2019).
24. Afanasyev, A.N., Voit, N.N. & Kirillov, S.Y.: Temporal Automata RVTI-Grammar for Processing Diagrams in Visual Languages as BPMN, eEPC and Askon-Volga. Proceedings of

the 2019 5th International Conference on Computer and Technology Applications - ICCTA 2019. Available at: http://dx.doi.org/10.1145/3323933.3324067 (2019).
25. Afanasyev, A., Voit, N., Ukhanova, M., Kirillov, S., & Bochkov, S.: Ontology-Based Organizational and Technical Components Semantic Model Development. Proceedings of the 2020 3rd International Conference on Geoinformatics and Data Analysis. doi:10.1145/3397056.3397089 (2020)