Grammatical-algebraic approach to control, analysis, diagnostics, modeling, interpretation and reengineering of dynamic design workflows (DDW) of a large design organization

A. Afanasev, N. Voit, M. Ukhanova, I. Ionova

Ulyanovsk state technical uni versity, Ulyanovsk, Russia
e-mail: {a.afanasev, n.voit}@ulstu.ru, mari-u@inbox.ru, epira@mail.ru

Abstract. The authors developed a new grammatical-algebraic approach to the analysis, control, diagnostics, modeling, interpretation and reengineering of DDW in the conditions of a large design organization in the part of design and technological preparation of production (DTPP)

1 Introduction

"Nowadays, business enterprises often need to dynamically reconfigure their internal processes in order to improve the efficiency of the business flow. However, modifications of the workflow usually lead to several problems in terms of deadlock freedom, completeness and security." [1]. Although, design, analysis, control and modelling of dynamic workflows are famous problem seeing flexible reaction to changing business process.

The dynamic development of complex automated systems is connected with the adaptation of workflows to changes in system requirements. "We consider enterprise/business agility as a property of an enterprise to function in the highly dynamic world" [2] and it has two features: 1) to adapt to changes in the environment; 2) to discover new opportunities constantly appearing in the dynamic world to launch completely new products (services). The agility implementation requires a new approach that allows designers to discover changes and opportunities of complex automated systems development, and to influence on them accordingly. The need to develop this approach appeared when the degree of changes in requirements of development increased. For example, in [3] the author says: "Industry and technology move too fast, requirements change at rates that swamp traditional methods."

Whitestein Technologies, Magenta Technologies, SkodaAuto, Volkswagen, Saarstahl AG note that the first generation of static management systems of product lifecycle and project workflows [4] can no longer meet the requirements of many companies, approach and automated tools of the first generation of project workflow standardization have already exhausted its resources, and as a result, there are poorly formalized (poor-quality) processes, stimulating the growth of expenses for their development and improvement.

The authors of this paper used the definition given in [5], for a dynamic workflow – workflow adjusted to the environmental changes. ProBis [6] is considered to be a traditional workflow management system. Dynamic project workflow management systems according to the works [7-9] include YAWL (Yet Another Workflow Language) and iPB.

According to the ISO 9000: 2000 standard, analysis, monitoring, diagnostics, modeling, interpretation and reengineering contain the verification and validation stages.

Verification is a confirmation on the basis of objective evidence that the requirements have been met.

Validation is a confirmation on the basis of objective evidence that the requirements intended for a particular use or application have been met.

2 Related works

2.1 Scientific schools and tools

Analysis and control of design workflows are carried out by the scientific schools of the Higher School of Economics, MSTU STANKIN, MVTU them. N.E. Bauman, UlSTU, POMI of the Russian Academy of Sciences, the Moscow State University of Economics, the Institute of System Programming of the Russian Academy of Sciences (Russia), the Carnegie Mellon University (USA), VERIMAG (France) laboratories, such scientists as Afanasyev A.N. [10-14], Karpov Yu.G. [15], Sosnin P.I., Lifshits Yu. [16], Yarushkina N.G., Kalyanov G.N. [17], Konev B. Yu., Shalyto A.A., Savenkov K., Kulyamin V. V. (Russia), as well as Neda Saeedloei, Gopal Gupta [18], Clark E.M., Buch G. (USA), Yuan Wang, Yushun Fan [19] (China), and Rational Unified Process (RUP) technologies [20], PBWD methodology, Unifyid Model Language (UML) modeling languages [21], extended Event Driven Process chain (eEPC), BPMN, IDEF0, IDEF3, Amber, Promela, YAWL, the Booch Methodology [22], Hierarchical Object Oriented Requirement Analysis (HOORA) [23], Jacobson Method [24], Object Modeling Technique (OMT) [25], Planguage [26], Shula-Mellor Object-Oriented Analysis Method [27], Software Cost Reduction Requirements Method (SCR) [28], Software Requirements Engineering Methodology (SREM) [29], Storyboard Prototyping [30], Structured Analysis and Design Technique (SADT) [31], as well as Structured Analysis and System Specification (SASS), Volere method, Win Win approach, Component-based methods (COTS-Aware Requirements Engineering (CARE), Off-the-Shelf Option (OTSO)) [32].

Karpov Yu.G. in [15] presents the Model Checking approach to the analysis, control, modeling and reengineering of business processes, in which the main drawback is the study of the model, and not the system itself, so the question arises about the adequacy of the model to the system, as well as the complexity of solving the problems listed above Is exponential.

Neda Saeedloei and Gopal Gupta [18] applied a temporal automaton realizing temporal context-sensitive grammar-grammar to the analysis of cyber-physical systems with the subsequent translation of this grammar into co-inductive logical programming for the Prolog interpreter.

Yuan Wang and Yushun Fan [19] suggest using the temporal logic of actions to describe the workflows in the graph form, which requires the description of all graph routes in the temporal logic of deutsviy formulas and the proof of the opposite assumption. The use of linear temporal logic to formalize the route from tasks, branches AND, OR and JOIN convergence, however, the question of the adequacy of constructing the description of workflows in the graph form remains unresolved.

The database of tools for analysis and control of cyber-physical systems, including work flows, is available at [33] and is presented in the table.

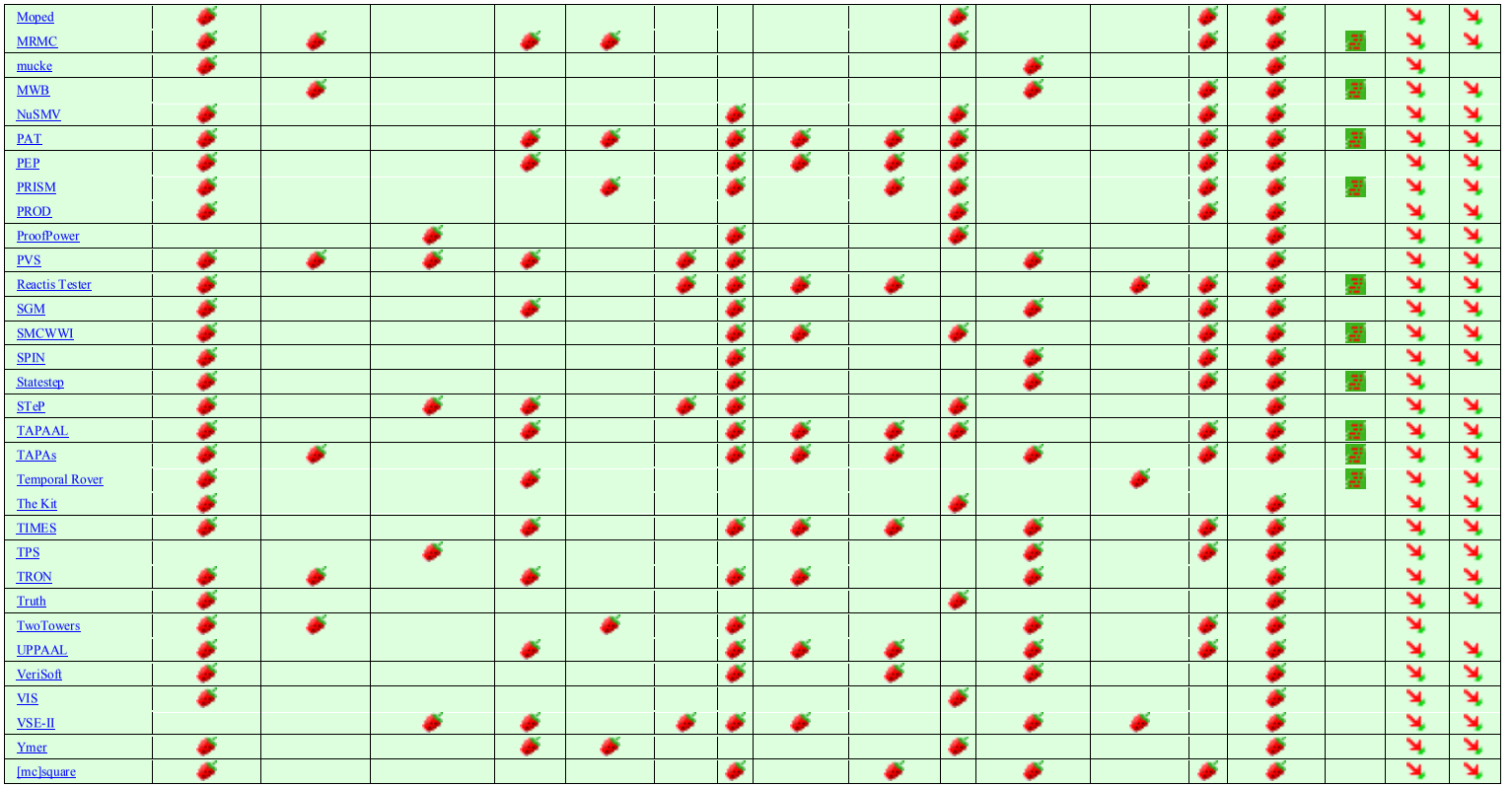
Moreover, the meaning of the symbols used in the table is the following: <NOTHING> is the tool doesn't have the feature; is the tool has the feature; is the tool has the feature and additional information is displayed, when the cursor is over the icon (if supported by the browser); is reference (e.g. e-mail or web site).

In addition, there are good tools CPN Tools [34], "Roméo - A tool for Time Petri Nets analysis" [35], TimesTool [36], Tina Toolbox [37], Visual Object Net ++ [38].

2.2 The analysis, the control and the diagnostic within a structurial conflict errors (Verification) [39]

The analysis and the control involve the checking of the syntactical correctness of a workflow model. In contrast to the context of programming languages where syntax only refers to the language, we incorporate in our notion of syntactical correctness both the structure and the behavior of the workflow model. Although workflow models that have been derived in the way as described in this section already implement some notions of syntactical correctness, models may be extended or changed by human intervention before they are considered complete. Typically, human errors in designing the workflow on the basis of the product data model may cause dead tasks, deadlocks, livelocks, etc. Especially when a workflow model becomes large, i.e., when it incorporates hundreds of tasks, it is difficult for human designers to oversee the complete model.

	Purpose <AND>			Specific Features <AND>			Graphical Interface <AND>			Availability <OR>			Platforms <OR>			Contact <AND>	
Name	Model Checking	Equiv. Checking	Theorem Proving	Real Time	Probabi listic	Hybrid	GUI	Graph. Specif.	Graph. Sim.	Free	Free Under Cond.	Commer cial	Win.	Unix & related	Others	Web Site	Email
ABC		✓								✓			✓	✓	✓	✓	✓
ACL2			✓								✓		✓	✓		✓	✓
Alpina	✓						✓	✓		✓			✓	✓		✓	✓
APMC	✓				✓		✓				✓			✓		✓	✓
ARC	✓								✓		✓			✓		✓	✓
Atelier B			✓				✓					✓		✓		✓	✓
Bandera	✓						✓	✓		✓			✓	✓		✓	✓
Blast	✓									✓			✓	✓		✓	✓
Cadence SMV	✓		✓				✓				✓		✓	✓		✓	✓
CADiZ			✓				✓			✓				✓		✓	✓
CADP	✓	✓			✓		✓		✓		✓		✓	✓	✓	✓	✓
CBMC							✓			✓			✓	✓		✓	✓
CWB - NC	✓	✓		✓			✓				✓		✓	✓		✓	✓
DBRover	✓			✓			✓	✓	✓			✓	✓	✓		✓	✓
DiVinE Tool	✓									✓				✓		✓	✓
DREAM	✓			✓						✓			✓	✓	✓	✓	✓
Edinburgh CWB	✓	✓									✓		✓	✓	✓	✓	✓
Expander2	✓	✓	✓			✓	✓		✓	✓				✓		✓	✓
Fc2Tools	✓	✓						✓	✓	✓				✓		✓	✓
FDR		✓					✓				✓	✓		✓		✓	✓
GEAR	✓						✓	✓	✓				✓	✓		✓	
HSolver	✓					✓				✓				✓		✓	✓
HyTech	✓			✓		✓		✓		✓			✓	✓		✓	✓
IF	✓			✓			✓		✓	✓			✓	✓		✓	✓
INA	✓									✓			✓	✓		✓	✓
IOA Toolkit			✓							✓			✓	✓	✓	✓	✓
isabelle			✓				✓			✓				✓	✓	✓	✓
JPF	✓									✓			✓	✓		✓	✓
KeY			✓				✓			✓			✓	✓		✓	✓
KRONOS	✓	✓		✓						✓			✓	✓		✓	
LP			✓							✓			✓	✓		✓	✓
LTSA	✓						✓		✓	✓			✓	✓		✓	✓
MCRL	✓	✓	✓						✓	✓				✓		✓	✓
mCRL2	✓	✓	✓	✓			✓		✓	✓			✓	✓	✓	✓	✓
MetaPRL			✓				✓			✓			✓	✓	✓	✓	✓
Mocha	✓	✓					✓		✓	✓			✓	✓		✓	✓


Moped
MRMC
mucke
MWB
NuSMV
PAT
PEP
PRISM
PROD
ProofPower
PVS
Reactis Tester
SGM
SMCWWI
SPIN
Statestep
STeP
TAPAAL
TAPAs
Temporal Rover
The Kit
TIMES
TPS
TRON
Truth
TwoTowers
UPPAAL
VeriSoft
VIS
VSE-II
Ymer
[mc]square

The analysis and the control of a workflow process for structural conflicts are a computationally complex problem and many approaches can be used to do this. However, the approach adopted for workflow analysis and control would correspond to the process definition language. Due to the computational complexity of the problem, only few approaches succeed in doing the workflow analysis and control under reasonable time limits for all kinds of workflow graphs. A discussion is given by Van der Aalst and Ter Hofstede (2000). Woflan tool has been created by H.W.M. Verbeek and W.M.P. Van der Aalst for an analysis and control the structural conflict errors in WF-nets. Apart from analysis and control structural conflict errors, Woflan can also be used for inheritance checking. Flowmake tool has been created by Wasim Sadiq and Maria E. Orlowska based on their Graph Reduction algorithm for analysis and control the structural conflict errors in Workflow graphs. Flowmake is based on the Graph Reduction algorithm by Wasim Sadiq and Maria E. Orlowska. This algorithm is not complete as it fails to check the analysis and control problems in special kind of workflow graphs called overlapped workflow graphs. It is necessary to have an analysis and control algorithm that checks all kinds of workflow graphs, as it is impossible to seclude overlapped graphs while designing business workflow processes. For this purpose, Hao Lin et al., defined an algorithm to analyse and control all kinds of workflow graphs.

Requirements verification is the process of ensuring, that requirements statements are accurate, complete and that they demonstrate the desired quality characteristics. (Wiegers, 1999)

2.3 The analysis, the control and the diagnostic within a semantical conflict errors (Validation) [39]

Perhaps the most important step in the evaluation phase is the analysis, the control and the diagnostic within a semantical conflict errors of the derived workflow designs by experts. The analysis, the control and the diagnostic involve the semantic correctness of the model: are the right things being done? Although the product specification is the proper source for deriving what should be done, misinterpretations or improper use of may be the cause of a faulty workflow design. Note that semantic correctness supposes a syntactic correctness of the workflow involved, as checked in the analysis, the control within a structurial conflict errors step.

From a system development point of view, it is important to validate a process design prior to the implementation of the workflow and the automation of processing steps. It is well known that design errors that are found late in the project are very costly to correct. Martin (1991) estimated that in software development finding a design error during the programming, testing, and maintenance phases is respectively 3, 10 and 100 times more costly than finding it during the design. From a change management perspective, it is also valuable to confront end-users with a design before further development takes place. This approach involves users in the design and it enables them to give feedback. It is also desirable that end-users realize that although the new process design may be structurally different from the process they are used to, it can be used for delivering the same type of products as before.

For all named the analysis, the control and the diagnostic purposes, there are different means available. Sommerville and Sawyer (1997) name formal inspections, developing draft manuals, paraphrasing, validation checklists, and prototyping. Casimir (1995) also names the gaming concept as a means for system design validation.

Common techniques to support validation include prototyping (executable model of the requirements), simulation and development of (mathematical) models [40].

The PBWD methodology is presented in Figure 1 [41].

Requirements validation is concerned with the process of examining the requirements document to ensure that it defines the right system (i.e. the system that the user expects). (Kotonya & Sommerville, 1998)

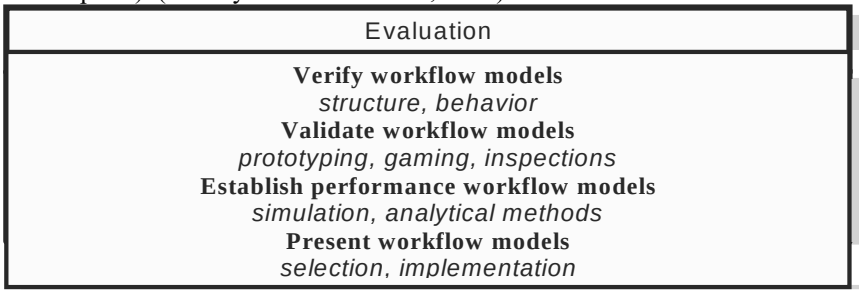


Figure 1. The PBWD methodology

3 Grammatical-algebraic approach to control, analysis, modeling, interpretation and reengineering of DDW

3.1 Structural-behavioral analysis, control and diagnosis of DDW (Verification)

Structural-behavioral analysis of DDW ensures control of its topological correctness and diagnostics (determination of the cause of the error in the form of a route in the DDW leading to an error) according to the following properties: attainability, safety, liveliness, justice [15].

The reachable property is the ability of the system (person) to obtain the desired result in the future, following the DDW.

Security properties (security) are both the absence of queues from tasks in the DDW when the enterprise's goal is achieved.

The liveness property is the activity of all structural units in the DDW, i.e. There is no such structure in DDW that can be removed, and the system will work without behavior changes.

The property of fairness is both a causal relationship of issuing tasks and obtaining resources for their implementation.

When analyzing, monitoring and diagnosing DDW, as a rule, the following series of typical structural errors are identified: deadlock, endless loop, problems with access to resources, blocking of resources, limitation of liveliness (Limiting the liveliness). In [39] they cluster them into two categories: The Lack of Synchronization structural conflict, the deadlock structural conflict.

The authors propose to use the developed ωRVT-grammar for stochastic-behavioral analysis, control and diagnosis of DDW.

3.2 Semantic analysis, control and diagnostics of DDW (Validation)

The semantic errors of the DDW are the most "dangerous" in terms of the consequences of incorrect system behavior, and also difficult to detect.

Validation, however, checks the final draft of the document for completeness, ambiguity and correctness. Demonstrating that a set of requirements meets the needs of the stakeholders. Common techniques to support validation include prototyping (executable model of the requirements), simulation and development of (mathematical) models.

Engineers-designers, software engineers often use the Error Checklist, which consists of listing the necessary test works to determine a malfunction in the system operation or a conflict situation in the DDW.

Here is an example of Checklist's mistakes of a software engineer, taken from [40].

1. Identify interfaces between the system and the operating environment in order to analyse hazards of the interfaces.
2. Identify interfaces between the system and the operating environment in order to analyse system level hazards.
3. Identify interfaces between the system and the operating environment in order to analyse human-machine interfaces.
4. Identify interfaces between the system and the operating environment in order to analyse hardware-software interfaces.
5. Identify interactions of subsystems within the system in order to analyse propagation of failure modes (i.e., hazards).
6. Identify interactions of hardware subsystems within the system in order to analyse incompatibilities between or at interfaces.
7. Identify interfaces between disciplines in order to interpret disciplinary specific terminology between disciplines.
8. Identify interfaces between disciplines in order to co-ordinate interdisciplinary efforts to mitigate risk.
9. Identify human-machine interfaces within the system operating parameters in order to assure that system demands do not exceed human physical or cognitive limitations. (Have we uncovered a potential fault)
10. Identify hardware-software interfaces within the system to determine compatibility of software and hardware at the interfaces. (Have we uncovered a potential fault)
11. Assess the adequacy of information exchange at hardware-software interfaces within the system.
12. Identify human-hardware-software-environment interfaces within the system to analyse the effect that the software has on the system.
13. Identify applicability of hardware failures as they affect the system.
14. Identify the effect of the environment on the system.
15. Identify the importance of design element factors in terms of their effect on the system.
16. Identify software functions that are critical to the safe operation of the system.
17. Identify safety critical elements in the software specification.
18. Discern proper and improper software controls for hardware operation.
19. Develop criteria to ensure that human limits and boundaries of operation are not exceeded.
20. Identify relationships between human operators, procedures, machines, and the environment which ensured total system operational safety.
21. Define relationships between human operators, procedures, machines, and the environment to ensure total system operational safety.
22. Evaluate workplace design to ensure that operational flows and human requirements are compatible with safe operations.
23. Develop and evaluate procedures to ensure proper operational flows and correct operator reactions to anomalies.
24. Define proper safety program activities commensurate with the development life cycle of the system.
25. Define and co-ordinate the system safety program life cycle with the system engineering life cycle.
26. Identify the safety implications associated with each life cycle phase of the system.
27. Identify the effect of the life cycle environments on the safety of the system.
28. Identify the effect of the system's life cycle on the environment.

29. Evaluate procedures to ensure proper operational flows and correct operator reactions to anomalies.
30. Determine potential human errors and their effects on the system.

The authors developed a method for locating and localizing semantic errors in DDW, based on temporal logics, automata and ωRVT-grammar.

3.3 Formalization of the DDW properties specification

The DDW properties specification describes the requirements for the behavior of the system (executor) and can be specified by the temporal logic formula in the form of a disjunctive-normal form as follows:

$$\varphi::p|\varphi|\neg\varphi|\varphi\vee\varphi|\circ\varphi|\varphi\cup\varphi, \quad (1)$$

where p is a temporal atomic predicate (atomic predicate), ¬ is a negation, ∨ is logical 'ИЛИ', an operator ∘ is a temporal operator NextTime (X), ∪ is a temporal operator Until. Moreover, eemporal atomic predicate p eakes on value '0' or '1' at specific time intervals and, unlike the predicate of first-order logic, does not depend on its structure.

Example. The chief designer sector checks design documentation (DD) for the product for errors, makes a decision: give it for revision in the presence of errors or, in the absence of errors, transfer to design engineers in separate design bureau – Design; and the head of the laboratory sector checks the design documentation (DD) for the product for errors, makes a decision: return it for revision in the presence of errors or, in the absence of errors, transfer it to the laboratory in the Laboratory. The formal entry looks like this:

$$\varphi = \text{Design} \vee \text{Laboratory} \cup (\text{No Errors in Problem Definition}), \quad (2)$$

where Design is a temporal atomic predicate: a sign of transfer of DD in Design, Laboratory is a temporal atomic predicate: a sign of transfer of DD to the Laboratory, (No Errors in Problem Definition) is a temporal atomic predicate: a sign of no errors in the DD.

To further simplify the temporal algebraic conclusions, we define φ through the temporal operator ° as:

$$\varphi = \text{NE} \vee (\text{D} \vee \text{L}) \wedge {}^{\circ}\varphi\,, \quad (3)$$

where NE is No Errors in Probmlem Definition, D is Design, L is Laboratory.

3.4 Definition timed ωRVT-grammars

The authors have developed automaton grammar, called ωRVT-grammar, to analysis and check (control) diagrams with time for the tools, described in the papers [10-14].

ωRVT-grammar has a basis of the L (R) language grammar, which can be written as:

$$G = \left(V, \Sigma, \tilde{\Sigma}, R, T, r_0\right), \quad (1)$$

where $V = \{v_l, l = \overline{1,L}\}$ is an auxiliary alphabet (the alphabetical operations with the internal memory, and $\text{card}(V) \to \infty$); $\Sigma = \{a_l, l = \overline{1,T}\}$ is a terminal alphabet, which is the union of its graphic objects and links (the set of primitives); $\tilde{\Sigma} = \{\tilde{a}_l, t =$

$\overline{1,\tilde{T}}\}$ is a quasi-terminal alphabet, extending the terminal alphabet. The alphabet includes:

- quasi-terms of graphic objects,
- quasi-terms of graphic objects with more than one input,
- quasi-terms of links, marked with the specific semantic;
- quasi-term for the end of an analysis;
- $R = \{r_i, i = \overline{1,I}\}$ is the scheme of the grammar (a set of rules, where each complex r_i consists a subset P_{ij} of rules, where $r_i = \{P_{ij}, j = \overline{1,J}\}$);
- $r_0 \in R$ is an axiom of RVT-grammar (the initial complex of rules), $r_k \in R$ is a final complex of rules.

The complex of rules $P_{lm} \in r_i$ is given as:

$$a_l^{[t_i]} \xrightarrow{W\gamma(v_1,\dots,v_n)} r_m \wedge \varphi, \tag{2}$$

where $\mathrm{W}_\gamma(v_1,\dots,v_\mathrm{n})$ – n-ary relation, defining an operation with the internal memory depending on $\gamma \in \{0, 1, 2, 3\}$; $r_m \in R$ is the receiver of rules, $T \in \{t_1, t_2, t_3 \dots, t_n\}$ is a set of timed labels (time stamp) with given functions $FT\Sigma: \Sigma \to T$ and $FT\tilde{\Sigma}: \tilde{\Sigma} \to T$ accordingly; φ ia a temporal formula of a specification.

The internal memory is presented by a stack for processing the graphic objects that have more than one output to save the information of link-marks, and elastic tapes for processing the graphic objects that have more than one input to mark the number of returns to a given vertex, and hence the number of incoming links. It should be noted that the elastic tape reads data from cells of the internal memory without a content destruction, and the cells of elastic tapes operates on data as a counter defined on positive integers.

The chain of $\varphi = \alpha_{l1}, \alpha_{l2}, \dots, \alpha_{l\lambda}$ is called RVT-derivation $\alpha_{l\lambda}$ from α_{l1} and it is denoted $\alpha_{l1} \overset{RV}{\Rightarrow} \alpha_{l\lambda}$ if for any $\xi < \lambda$ and $r_e \in R$ are as $\alpha_{l\xi+1} \in r_e$, $\left(a_l^{[t_i]} \xrightarrow{W\gamma(\gamma_1,\dots,\gamma_n)} r_e \wedge \varphi\right) \in r_i$.

ωRVT-grammar is effective both for generating and recognizing.

The application of any complex of rules r_0 (ωRVT-grammar axiom) generates some chain of language L on its ωRVT-grammar. The complex of rules determines both the initial symbol of generated chain, the operation on the internal memory, and also the name of receiver of rules. The generation is completed using a complex of rules with r_k on the right-hand side.

The recognition of the chain runs verifying the first symbol using rules r_0, while next symbol appearing, and the last symbol of the chain must belong to a complex of rules with r_k on the left-hand side.

The use of rules is accompanied with appropriate operations on the internal memory. The internal memory is empty at the start, and at the end of these processes, the memory contains operations of rules with r_k on the right-hand side.

Temporal formula φ says that the implication of the contents of the magazine belt $(v_1,\dots,v_n)$ must satisfy φ, otherwise, there is a semantic error in the DDW, i.e.

$$\exists\varphi \to w_2(v_1,\dots,v_n)/w_3(v_1,\dots,v_n \neq \emptyset) \tag{3}$$

Taking into account formula (3) and slaves [10-19, 32, 39, 40], the authors introduce the following new class of semantic errors of DDW: lack of necessary (for example, lack of quality control of the source code of the computer program due to the short development time in DDW, which can lead to costly mistakes in implementation, implementation and operation).

Thus, the DDW diagnostics is realized by means of a step-by-step trace of the content of the magazine tape with the fulfillment of the condition φ.

3.6 Interpreting and reengineering DDW

The implementation part of the DDW in the form of the source code of the graphic primitive, which is an object of the compiler interpretation, can be represented in different languages, including high-level languages (for example, Object Pascal (Delphi), Visual Basic for Application), which allows it to be converted in the diagram view, including Active Diagram UML, and check for errors using ωRVT-grammar.

Exploitation DDW allows to present in detail the design and technological processes of the enterprise and can serve as useful information to the management with the aim of reorganization of the enterprise, including reengineering. When reengineering DDW, the existing workers (operating in production) are analyzed, and not planned in the future. At the same time, the structural and behavioral analysis of the DDW, the semantic analysis of the DDW, the introduction of changes in the DDW with the aim of optimizing them in terms of greater automation of design and technological work and increasing the overall production efficiency are carried out. The optimization criteria can include such categories as time, number, volume, money, etc., which can be verified automatically and validated using the temporal formula φ present in the mathematical apparatus ωRVT-grammar.

4 Conclusion

The authors developed a grammatical-algebraic approach to the analysis, control, diagnostics, modeling, interpretation and reengineering of DDW based on its own ωRVT-grammar. The scientific significance of the approach is represented by the ωRVT-grammar, which takes into account the temporal nature of the DDW, provides structural-behavioral and semantic analysis, control, diagnostics, modeling, interpretation and reengineering of the DDW, and extends the class of errors.

In the future works, the authors plan to conduct experiments on BPMN, eEPC, Active Diagrma UML and other graphical temporal languages for the development of complex systems, taking into account the data of design and technological preparation for the production of a real enterprise.

Acknowledgment

This research is supported by the grant of the Ministry of Education and Science of the Russian Federation, the project № 2.1615.2017/4.6. The reported study was funded by RFBR and Government of Ulyanovsk Region according to the research project № 16-47-732152. The reported study was funded by RFBR according to the research project № 17-07-01417.

References:

1. Juan Carlos Polanco Aguilar, Koji Hasebe, Manuel Mazzara, and Kazuhiko Kato, "Model Checking of BPMN Models for Reconfigurable Workflows," https://www.researchgate.net/publication/304788360_Model_Checking_of_BPMN_Models_for_Reconfigurable_Workflows

2. Bohdana Sherehiy, Waldemar Karwowski, John K Layer, "A review of enterprise agility: Concepts, frameworks, and attributes," International Journal of Industrial Ergonomics, 37, pp. 445-460, 2007. http://www.sciencedirect.com/science/article/pii/S0169814107000236

3. Highsmith J., Orr K., Cockburn A., "Extreme programming", in: E-Business Application Delivery, Feb. 2000, pp. 4–17.

4. A global Swiss company offering advanced intelligent application software for multiple business sectors, 2016, http://whitestein.com/

5. Ilia Bider, Amin Jalali, "Agile Business Process Development: Why, How and When Applying Nonaka's theory of knowledge transformation to business process development. Information Systems and e-Business Management," Springer-Verlag Berlin Heidelberg, 2014. https://www.researchgate.net/publication/266078141

6. T. Andersson, A. Andersson-Ceder & I. Bider, "State flow as a way of analyzing business processes-case studies. Logistics Information Management," 15(1), pp.34-45, 2002.

7. YAWL Foundation, YAWL. 2004, http://www.yawlfoundation.org/

8. I. Bider, "Analysis of Agile Software Development from the Knowledge Transformation Perspective," Proceedings of the 13th International Conference on Perspectives in Business Informatics Research (BIR 2014). Lund, Sweden. Springer, LNBIP, pp. 143-157, 2014.

9. IbisSoft, "iPB Reference Manual," 2009. http://docs.ibissoft.se/node/3

10. Alexander Afanasyev, Nikolay Voit, "Intelligent Agent System to Analysis Manufacturing Process Models," Proceedings of the First International Scientific Conference «Intelligent Information Technologies for Industry» (IITI'16) vol.451 of the series Advances in Intelligent Systems and Computing. Russia, pp. 395-403 (2016)

11. Alexander Afanasyev, Nikolay Voit, Rinat Gaynullin, "The Analysis of Diagrammatic Models of Workflows in Design of the Complex Automated Systems," Proceedings of the First International Scientific Conference «Intelligent Information Technologies for Industry» (IITI'16) vol. 450 of the series Advances in Intelligent Systems and Computing. Russia, pp. 227-236 (2016)

12. A.N. Afanasyev, N.N. Voit, R.F. Gainullin, "Diagrammatic models processing in designing the complex automated systems," 10th IEEE International Conference on Application of Information and Communication Technologies (AICT). Baku, Azerbaijan, pp. 441-445 (2016)

13. A.N. Afanasyev, N.N. Voit, E.Yu. Voevodin, R.F. Gainullin, "Control of UML diagrams in designing automated systems software," Proceedings of the 9th IEEE International conference on Application of Information and Communication Technologies: AICT – 2015, pp. 285-288 (2015)

14. A.N. Afanasev, N.N. Voit, E.Yu. Voevodin, R.F. Gainullin, "Analysis of Diagrammatic Models in the Design of Automated Software Systems," Object Systems – 2015: Proceedings of X International Theoretical and Practical Conference (Rostov-on-Don, 10-12 May, 2015) / Edited by Pavel P. Oleynik. – Russia, Rostov-on-Don: SI (b) SRSPU (NPI), pp. 124-129 (2015)

15. Карпов Ю. Г. MODEL CHECKING. Верификация параллельных и распределенных программных систем. – СПб.: БХВ-Петербург, 2010. – 560 с. (in russia)
16. Верификация программы и темпоральные логики. URL: http://logic.pdmi.ras.ru/~yura/modern/034.pdf (in russia)
17. Калянов Г.Н. Моделирование, анализ, реорганизация и оптимизация бизнес-процессов / Учебное пособие. — М.: Финансы и статистика, 2006. — 240 с. URL: http://www.twirpx.com/file/2204790/ (in russia)
18. Saeedloei, Neda, and Gopal Gupta. "Timed definite clause omega-grammars." LIPIcs-Leibniz International Proceedings in Informatics. Vol. 7. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2010.
19. Wang, Yuan, and Yushun Fan. "Using temporal logics for modeling and analysis of workflows." E-Commerce Technology for Dynamic E-Business, 2004. IEEE International Conference on. IEEE, 2004.
20. Kruchten, P. 1998. The Rational Unified Process. Addison-Wesley.
21. Booch, G.; Jacobson, I. & Rumbaugh, J. 1998. The Unified Modeling Language User Guide. Addison-Wesley.
22. Booch, G. 1994. Object-oriented Analysis and Design with Applications, 2nd edition. Addison-Wesley.
23. HOORA's homepage. URL: http://www.hoora.org/
24. Jacobson, I.; Christerson, M.; Jonsson, P., & Gunnar, O. 1992. Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley.
25. Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F. & Lorensen, W. 1991. Object-Oriented Modeling and Design. Prentice Hall.
26. Gilb, T. 2003. Competitive Engineering. Addison-Wesley.
27. Shlaer, S. & Mellor, S. 1988. Object-Oriented System Analysis: Modeling the World in Data. Yourdon Press Computing Series. Prentice-Hall.
28. Parnas, D.L. 1978. Software Requirements for the A-7 Aircraft. Technical Report 3876. Naval Research Laboratory. Washington DC.
29. Alford, M. 1977. A Requirements Engineering Methodology for Real Time Processing Requirements. IEEE Transaction on Software Engineering. Vol. 3. No.1. Pages 60−69.
30. Andriole, S. 1989. Storyboard Prototyping for Systems Design: A New Approach to User Requirements Analysis. Q E D Pub Co.
31. Marca, D.A. & McGowan, C.L. 1988. SADT: Structured Analysis and Design Techniques. McGraw-Hill.
32. Parviainen, Päivi, et al. "Requirements engineering inventory of technologies." VTT PUBLICATIONS (2003).
33. YAHODA.URL: http://web.archive.org/web/20120220001353/http://anna.fi.muni.cz/yahoda/
34. CPN Tools. URL: http://cpntools.org/
35. Roméo. URL: http://romeo.rts-software.org
36. TimesTool. URL: http://www.timestool.com/documentation.shtml
37. Tina Toolbox. URL: http://projects.laas.fr/tina/
38. Visual Object Net++. URL: https://www.techfak.uni-bielefeld.de/~mchen/BioPNML/Intro/VON.html
39. Workflow Handbook 2005:/ Layna Fischer (editor), Published by Future Strategies Inc., Book Division.
40. Sofia Guerra, Adelard «Requirements Engineering Best Practice Guide for Refurbishment».
41. Hajo A. Reijers «Design and Control of Workflow Processes».