# Processing of Conceptual Diagrammatic Models Based on Automation Graphical Grammars

Alexander Afanasyev, Anatoliy Gladkikh, Nikolay Voit, and Sergey Kirillov(✉)

Ulyanovsk State Technical University, Ulyanovsk, Russia
{a.afanasev, n.voit}@ulstu.ru, kirillovsyu@gmail.com

**Abstract.** The presented work is devoted to conceptual design in area of creation and processing of diagram models. Authors offer a family of automatic graphical RV-grammars which allows to analyze, control and interpret diagrammatic models of automated systems presented in the widely used visual languages as UML, IDEF, BPMN. The estimation of time costs of the analysis is executed and the graph is constructed.

**Keywords:** Conceptual design · Automated systems · Visual languages
Diagrammatic models · Workflows · Automaton grammars

## 1 Introduction

Diagrammatic models developed in visual languages (UML, IDEF, eEPC, BPMN, SharePoint, ER, DFD, etc.) are actively used in the practice of designing complex automated systems (CAS), especially at conceptual stages. CAS includes, among others, e-learning environments. In particular, the open technology platform for distance learning Moodle contains more than 1.5 million lines of source code. For example, in master technology of RUP (Rational Unified Process) [1], UML is the basis for representation of architectural and software solutions, and when using the ARIS (Architecture of Integrated Information Systems) - eEPC. The use of diagram models allows to increase the efficiency of design solutions, to avoid "expensive" mistakes, to improve the understanding of the project by executors, to organize interaction between customers and project executors, to improve project documentation and in some cases to automate the process of obtaining program code. The solution of these problems is connected with the problem of "success" in the creation of CAS. According to research by Standish Group [2], only about 40% of projects meet the requirements of the declared functionality, implemented within the specified time and within the specified budget.

Due to the specifics of modern CAS design, the considered models are dynamic and distributed, which is determined by the changing business processes and the work of remote teams of designers in the environment.


A. Abraham et al. (Eds.): IITI 2018, AISC 875, pp. 369–378, 2019.
https://doi.org/10.1007/978-3-030-01821-4_39

The research of structural (primarily topological features) and semantic (connectivity of diagrams, possibly presented in various visual languages, in terms of the text component) is an actual task, which has great practical value.

## 2 Related Works

In the modern theory of graphic visual languages, a logical model is used to represent diagrams, which contains graphical objects and connections between them. To process such models, graphic grammars are used. John L. Pfaltz and Azreil Rosenfeld offered web grammar [3]. Zhang and Costagliola [4, 5] developed a positional graphical grammar related to context-free grammar. Wittenberg and Weitzman [6] developed a relational graphical grammar. Zhang and Orgun [7] described a graphical grammar preserving in their works. However, the mentioned graphical grammars have the following disadvantages:

1. Positional grammars. It doesn't involve the use of connection areas and cannot be used for graphical languages, whose objects have a dynamically changing number of inputs/outputs,
2. The authors of the relational grammars tells about the imperfection of the neutralization mechanism, specifically the incompleteness of the generated list of errors.
3. There is no control of semantic integrity (text attributes of complex diagram models presented in different visual languages), as well as semantic consistency in terms of structural issues of diagrams between themselves and the conceptual model as a whole.
4. The common disadvantages of the above grammars are: the increase in the number of products in the construction of grammar for unstructured graphic languages (with a constant number of primitives of the graphical language to describe all variants of unstructured there is a significant increase in the number of products), the complexity of grammar construction, the large time costs of analysis (analyzers built on the basis of the considered grammars, have a polynomial or exponential time of graphic diagrams analysis.

In [8], colored Petri nets are used for the dynamic semantic analysis of workflows, and in [9] the pi-Calculus approach formalizing workflows into algebraic statements of first-order logic. However, there is no analysis of the text component of the diagrammatic models in these methods.

In the most common tools for creating and processing diagram models, such as Microsoft Visio [10], Visual paradigm for UML [11], Aris Toolset [12], IBM Rational Software Architect (RSA) [13] analysis and control of diagram models is performed by direct methods, requires several "passes" depending on the type of error. There is no control over the structural features of the complex diagram models, and also the semantic control of the integrity and consistency of the structural and text attributes of the associated diagrammatic models of dynamic workflows.

## 3 Problem

Diagrammatic models of the conceptual design of automated systems have a number of features, the analysis of which allows us to identify a number of mistakes related to the most "expensive" ones.

From the structural (syntactic) point of view, these visual languages have a number of graphic objects with a variable number of inputs/outputs, through which complex structures associated with the use of "AND", "OR", "XOR" splitters can be organized. Splitters can "stand" at a considerable distance from each other, forming structures with a remote context. The methods and tools discussed above are not in a position to control such features. The results may include diagrams that contain freezes, deadlocks and deadlocks.

The next structural feature of the considered class of models is their complexity. This is especially true for UML and IDEF. For example, the top of the UML usage diagram can be deployed into an activity diagram, the individual blocks of which can correspond to class diagrams, etc. Implementing such a hierarchical dynamic end-to-end analysis is an important task.

Let us consider the semantic features of the diagrammatic models. Diagrams should not contain conflicting information.

Contradiction of the model can cause serious problems in its implementation and subsequent use in practice. For example, the presence of closed paths when mapping the aggregation or composition relationship leads to errors in the code that will implement the corresponding classes. Having elements with the same name and different property attributes in the same namespace also leads to ambiguous interpretations and can be a source of errors.

In the collective development of automated systems, the work of designers opens up additional sources of difficult to diagnose, distributed over a multitude of error diagrams.

When working in a team, it is important to monitor the consistency of concepts used in complex diagrams. Semantic control of complex diagrams allows you to monitor errors distributed across different diagrams. With semantic control, the complex idea created about an automated system becomes consistent and consistent. Early diagnostics of such errors can reduce development time by reducing the iterations associated with their correction.

Thus, the solution of these problems is related to the development of mechanisms that allow analysis and control of the syntactic and semantic features of conceptual models. The article proposes a family of automatic graphical grammars that allow efficient processing of such models.

## 4 RV-Grammar

RV-grammar of the language L (G) is the ordered five non-empty sets

$$G = \left(V, \Sigma, \widetilde{\Sigma}, R, r_0\right) \tag{1}$$

where $V = \left\{v_e, e = \overline{1,L}\right\}$ is auxiliary alphabet; $\Sigma = \left\{a_t, t = \overline{1,T}\right\}$ is terminal alphabet graphic language; $\widetilde{\Sigma} = \left\{\widetilde{at}, t = \overline{1,T}\right\}$ is quasi terminal alphabet; $R = \left\{r_i, i = \overline{0,I}\right\}$ is grammar G schema (set of names of products complexes, each complex $r_i$ consists of a subset $P_{ij}$ of products $r_i = \left\{P_{ij}, j = \overline{1,J}\right\}$; $r_0 \in R$ is RV-axiom grammar.

RV-grammar is automate-based grammar. It operates with inner memory to control correctness of diagrams. Finite state machine is used to choose rule by memory changing. A number of different structures are used as memory: stacks, lists, queries, etc. and combination of them. These conditions make possible to construct grammar with linear dependence from element count. All operation takes limited time, because reading and writing data is constant time operation. In each tack of diagram analysis operate with only one element. On Fig. 1 is presenter RV-grammar in graph form.

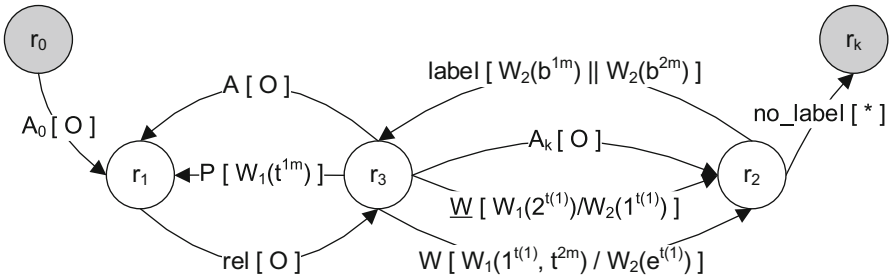


**Fig. 1.** RV - grammar graph form for the PGSA language after minimization

## 5 Hierarchical RVH-Grammar

The construction of a hierarchical RV grammar is based on grammatical models in the form of a recursive Woods network. Suppose that not only terminal (quasi terminal) symbols are allowed as symbols marking the arcs of the graph of RV grammars, but also nonterminal symbols denoting complex syntactic constructions whose presence is necessary so that the transition along the arc to the next complex is permissible. As nonterminal symbols, the names of production complexes can act, and as complexes themselves - other RV-grammars (we will call them subgrams). To organize work on the hierarchical RV grammar, a store mechanism is used, similar to that used when calling subroutines from the main program and then returning to the main program. Using RVH grammars allows you to reduce the amount of grammar, it is easy to modify it, effectively construct a grammar from existing subgrams (Fig. 2).

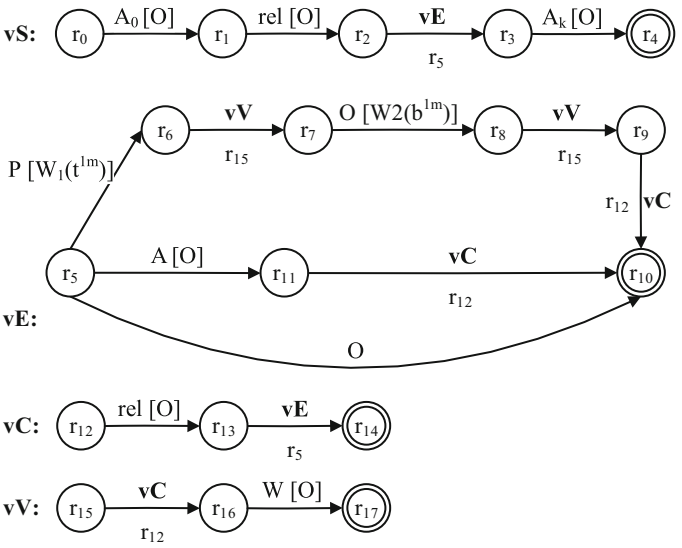


**Fig. 2.** Hierarchical RVH-grammar example

## 6 Temporal RVT-Grammars

Flows of design works are a powerful tool for analyzing the business processes of the enterprise and contain design tasks that are addressed to specific departments and executors of the enterprise.

Such design works can be performed simultaneously in different structural divisions and different executors, so the issues of synchronization, resource blocking, deadlocks, bottlenecks, etc., encountered in the theory of informatics, arise in the field of business process management enterprise. Such streams of design work can be represented in the temporal basis of "Before", "During", "After", putting the time schedule for the execution of the enterprise's work in accordance with the parameter.

Temporal RVT - grammar of a language L (G) is an ordered eight non-empty sets

$$G = \left(V, \Sigma, \widetilde{\Sigma}, C, E, R, T, r_0\right) \tag{2}$$

where $V = \left\{v_e, e = \overline{1,L}\right\}$ is auxiliary alphabet; $\Sigma = \left\{a_t, t = \overline{1,T}\right\}$ is terminal alphabet graphic language; $\widetilde{\Sigma} = \left\{\widetilde{at}, t = \overline{1,T}\right\}$ is quasi terminal alphabet; C is set of clock identifiers; E is the set of temporal relations "Before", "During", "After" (initialization of the clock $\{c := 0\}$, relations of the form $\{c \sim x\}$, where x the variable (the identifier of the clock), a is a constant, $\sim \in \{=, <, \leq, >, \geq\}$); $R = \left\{r_i, i = \overline{0,I}\right\}$ is grammar G schema (set of names of complexes of products, each complex $r_i$ consists of a subset $P_{ij}$ of products $r_i = \left\{P_{ij}, j = \overline{1,J}\right\}$); $T \in \{t_1, t_2, \ldots, t_n\}$ is a set of time stamps; $r_0 \in R$ is RV-axiom grammar.

RVT-grammar was developed for basic elements of BPM notation. Table form of this grammar presented in Table 1.

**Table 1.** Grammar example for simple BPMN diagram

| N | State | Quasi term | Next state | Operation with memory |
|---|---|---|---|---|
| 1 | r0 | A0 | r1 | o |
| 2 | r1 | rel | r3 | o |
| 3 | r2 | labelEG | r3 | $W_2(b^{1m}, b^{t(6)})$ |
| 4 | | labelPG | r3 | $W_2(b^{2m}, b^{t(6)})$ |
| 5 | r3 | Ai | r1 | o |
| 6 | | Aim | r1 | o |
| 7 | | Ait | r1 | $W_1(t_s^{t(6)})$ |
| 8 | | Akl | r2 | $W_3(e^{1m}, e^{2m})$ |
| 9 | | Ak | r4 | o |
| 10 | | A | r1 | $W_1(t_s^{t(6)})$ |
| 11 | | EGc | r1 | $W_1\left(t^{1m^{(n-1)}}\right)/W_3(k = 1)$ |
| 12 | | EG | r2 | $W_1(1^{t(1)}, k^{t(2)})/W_3(e^{t(2)}, k\ !\ = 1)$ |
| 13 | | _EG | r2 | $W_1(inc(m^{t(1)})/W_3(m^{t(1)} < k^{t(2)})$ |
| 14 | | _EGe | r1 | $W_1\left(t^{1m^{(n-1)}}\right)/W_3\left(m^{t(1)} = k^{t(2)}, p! = 1\right)$ |
| 15 | | _EGme | r1 | $o/W_3(m^{t(1)} = k^{t(2)}, p = 1)$ |
| 16 | | PGf | r1 | $W_1\left(t^{2m^{(n-1)}}\right)/W_3(k = 1)$ |
| 17 | | PG | r2 | $W_1(1^{t(3)}, k^{t(4)})/W_3(e^{t(3)}, k\ !\ = 1)$ |
| 18 | | _PG | r2 | $W_1(inc(m^{t(3)})/W_3(m^{t(3)} < k^{t(4)})$ |
| 19 | | _PGe | r1 | $W_1\left(t^{2m^{(n-1)}}\right)/W_3\left(m^{t(3)} = k^{t(4)}, p! = 1\right)$ |
| 20 | | _PGje | r1 | $W_1\left(t^{2m^{(n-1)}}\right)/W_3\left(m^{t(3)} = k^{t(4)}, p = 1\right)$ |
| 21 | r4 | no_label | r5 | * |
| 22 | r5 | | | |

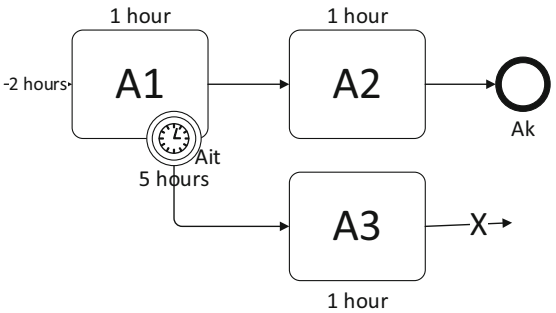


**Fig. 3.** Temporal BPMN diagram example

The example of diagram, which can be analyzed by RVT-grammar presented in Fig. 3. The main feature is time labels for every object from terminal alphabet.

## 7 The Method for Controlling the Semantic Integrity of Flow Diagrams

The control of the semantic integrity of the models under consideration is connected with the diagnosis of the consistency and consistency of the text component of the diagrams. An ontological approach is proposed for these purposes. Ontological analysis can be carried out during the process of analyzing the diagram or in the complete construction of an ontology upon completion of the analysis of the diagram. In the first case, it becomes possible to immediately point to a specific element that led to an error. However, you will have to spend more time, since ontology analysis should be performed on each element. In the second case, the process of combining ontologies must be carried out only once. Next, the second option will be considered. The general scheme of the ontological analysis of the diagrams is shown in Fig. 4.

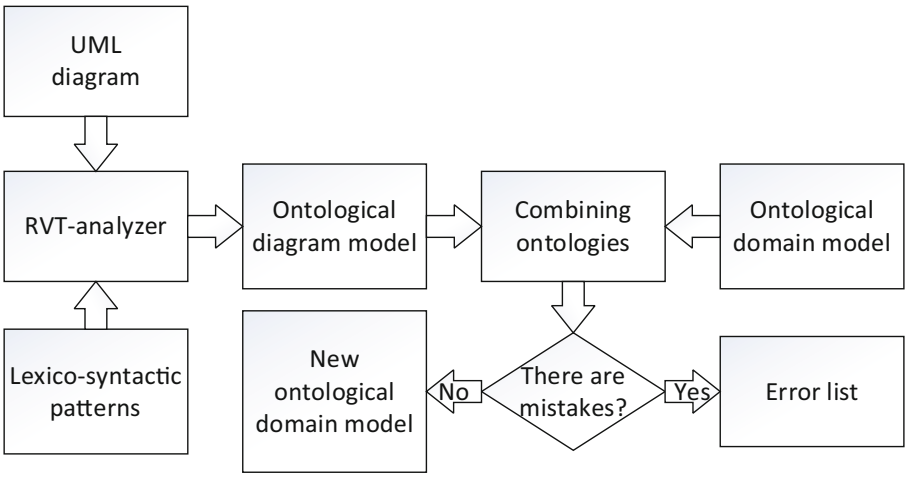


**Fig. 4.** Main scheme of diagram ontological analysis

To control the semantic integrity of different types of diagrams, we will make changes to the RV-grammar rule:

$$a_t(\chi,\varsigma) \xrightarrow{W_v(\gamma_1,\ldots,\gamma_n)} r_m \tag{3}$$

where $\chi$ is the procedure for extracting semantic information, and $\varsigma$ is the procedure for retrieving the temporal information. The procedure of $\chi$ is to find the appropriate rule in the list. The rule consists of two parts - replacement and pattern. The pattern part describes the lexico-syntactic pattern.

The replacement part specifies the location of this text unit in the partial semantic tree of the diagram.

## 8 Time Estimation of RV-Grammars

The RV analyzer has a linear time characteristic of the control, i.e. The time spent analyzing the input list of chart elements is determined by the linear function of its length and is calculated by the formula:

$$t = c * L_s \tag{4}$$

or

$$t = c * k * L_k, k = L_s * L_k \tag{5}$$

Where $L_s$ - the number of transitions over the states of the automaton, $L_k$ - the number of elements of the diagram and C - the algorithm implementation constant, which shows how many commands (operators) are spent on analyzing one object when implementing a control algorithm on a specific PC.

An experiment was conducted in which 500 PGSA diagrams were generated, then for each of them the number of generated elements was determined and the number of states of the analyzing machine was calculated. The result of the experiment is presented in the form of a dot diagram in Fig. 4, it confirms the linear dependence of the number of applications of grammar rules on the number of graphic objects and links. In addition, it was determined the expectation of the coefficient of equal and average deviation (Fig. 5).

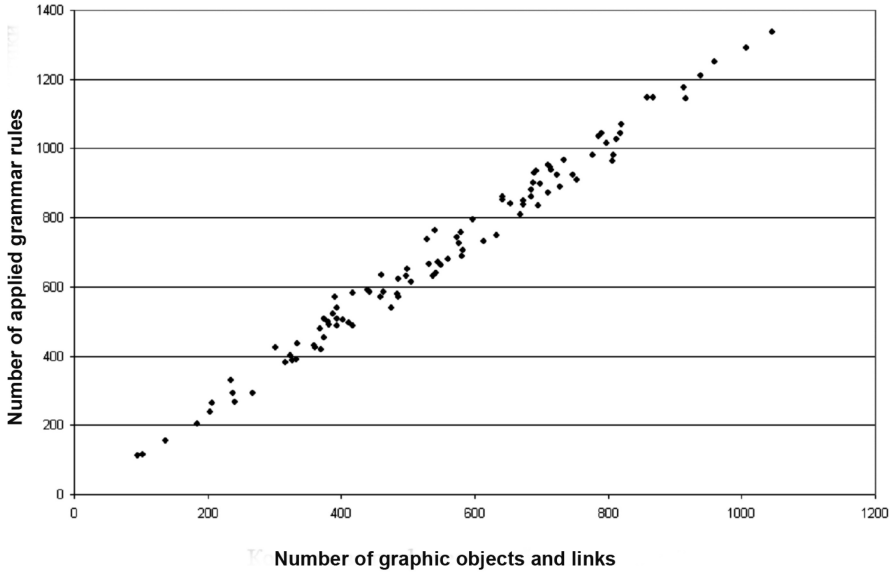


**Fig. 5.** The values of the experimental linearity coefficients for the PGSA language

## 9 Conclusion

A family of automatic graphical grammars has been developed that make it possible to extend the class of diagnosed errors in the process of developing conceptual models of automated systems. RVN - a grammar offering mechanisms for neutralizing errors, RVH is a grammar that assumes its construction on the basis of sub grammars, as well as an easy modification of the basic grammar and re-use of sub grammars, and RVTI is a grammar allowing to take into account time characteristics. Developed analyzers for most visual languages in the form of plug-ins for MS Visio.

Further directions of work are the expansion of the possibilities of semantic analysis of diagrammatic models from the point of view of coordinating text attributes of diagrams with project documentation.

**Acknowledgment.** The reported study was funded by RFBR according to the research project № 17-07-01417. The research is supported by a grant from the Ministry of Education and Science of the Russian Federation, project No. 2.1615.2017/4.6. The reported 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. Fowler, M.: New Programming Methodologies. http://www.maxkir.com/sd/newmethRUS.html. Accessed 14 May 2018
2. The Standish Group. http://www.standishgroup.com/outline. Accessed 14 May 2018
3. Fu, K.: Structural Methods of Pattern Recognition. Mir, Moscow (1977)
4. Zhang, D.Q., Zhang, K.: Reserved graph grammar: a specification tool for diagrammatic VPLs. In: IEEE Symposium on Visual Languages, Proceedings, pp. 284–291 (1997)
5. Costagliola, G., Lucia, A.D., Orece, S., Tortora, G.: A parsing methodology for the implementation of visual systems. http://www.dmi.unisa.it/people/costagliola/www/home/papers/method.ps.gz. Accessed 14 May 2018
6. Wittenburg, K., Weitzman, L.: Relational grammars: theory and practice in a visual language interface for process modeling. http://citeseer.ist.psu.edu/wittenburg96relational.html. Accessed 14 May 2018
7. Zhang, K.B., Zhang, K., Orgun, M.A.: Using graph grammar to implement global layout for a visual programming language generation system. In: Feng, D., Jin, J., Eades, P., Yan, H. (eds.) Conferences in Research and Practice in Information Technology, vol. 11, pp. 115–122. Australian Computer Society, Sydney (2002)
8. Jalali, A., Wohed, P., Ouyang, C.: Aspect oriented business process modelling with precedence. In: Mendling, J., Weidlich, M. (eds.) Business Process Model and Notation. BPMN 2012. Lecture Notes in Business Information Processing, vol. 125, pp. 23–37. Springer, Heidelberg (2012)
9. Hasso Plattner Institut. http://bpt.hpi.uni-potsdam.de. Accessed 14 May 2018
10. Roth, C.: Using Microsoft Visio 2010. Pearson Education, United States of America (2011)
11. Paradigm V. Visual paradigm for UML. Hong Kong: Visual Paradigm International. http://www.visual-paradigm.com/product/vpuml/. Accessed 14 May 2018

12. Santos Jr., P.S., Almeida, J.P.A., Pianissolla, T.L.: Uncovering the organisational modelling and business process modelling languages in the ARIS method. Int. J. Bus. Process Integr. Manag. **2**(5), 130–143 (2011)
13. Hoffmann, H.P.: Deploying model-based systems engineering with IBM® rational® solutions for systems and software engineering. In: Digital Avionics Systems Conference (DASC), 2012 IEEE/AIAA 31st. – IEEE, 2012. – C. 1–8 (2012)
14. Sharov, O., Afanasev, A.: Syntactically - implementation-oriented graphical language based on automatic graphical grammars. Programming **6**, 56–66 (2005)