Analysis and control of dynamic distributed workflows in the design and reengineering of complex automated systems*

A.N. Afanasyev, N.N. Voit, A.G. Igonin
«Computing technique» department
Ulyanovsk State Technical University
Ulyanovsk, Russia
{a.afanasev, n.voit}@ulstu.ru, igonin@ritg.ru

***Abstract*—Authors propose a method for analyzing, control and translation distributed workflows based on own RV-, RVTt-grammars. The languages BPMN and BPEL are used by the authors as an example. An estimation of the method effectiveness is presented.**

Keywords—diagrammatic workflow; grammars; analysis; translation

I. INTRODUCTION

Dynamic distributed workflows can be defined as a set of related operations, the implementation of which is aimed at achieving the company's goal, and they are active in solving project tasks and reengineering tasks. Such activities can come from a person, device or program.

Each company is created to fulfill certain goals, therefore in any company there are workflows, even if they are not described and not documented. The introduction of a formal description of project workflows allows companies to build company's quality management systems, to solve problems of building an effective management structure, to optimize activity based on key indicators.

Whitestein Technologies, Magenta Technologies, SkodaAuto, Volkswagen, Saarstahl AG note that the first generation of static management systems of product lifecycle and project workflows [1] 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 research work used to use the definition given in [2] for the dynamic project workflows – project workflows, adapted to changes in the environment. Dynamic distributed workflows are not necessarily the sequence of designer's actions. Dynamic distributed workflows can be implemented by several designers within the same division of the company, covering several of its divisions or even several divisions of different companies. Therefore, dynamic distributed workflows are represented as the parallel activity of many performers that consistently perform their functions.

In the case of a set of distributed project workflows within a single company or within the framework of associations of different companies, such flows must be monitored, analyzed and translated into relevant representations.

The management systems for such workflows are represented by frameworks that provide the accumulation of flows, their scheduling and presentation in the form of various graphical notations, e.g., SRML, BPMN, EPC, UML, IDEF0, WPDL, B2B [3].

The business processes representation as diagrams is intended to help designers to develop and analyze project decisions, designing preproduction, preproduction engineering with the help of logical reasoning on specific complex business processes. Many systems for controlling the project workflows are developed for different paradigms and have a scientifically closed research and application [4]. The methods for processing dynamic distributed workflows are mainly divided into supporting and not supporting dynamic properties [5]. Methods based on graphical languages UML, WPDL, BPMN cannot handle dynamic properties, and Petri networks, π-calculus, model checking are able to handle dynamic properties of dynamic distributed workflows.

The paper has the following structure. Section 2 presents the research problem. Section 3 briefly presents an overview of orchestration, choreography and the translation purpose. Section 4 contains RV-, RVTt-grammars for analysis, translation of dynamic distributed workflows. Section 5 presents the effectiveness assessment of the RV-grammar use. The conclusions and further research directions are presented in the last section.

II. PROBLEM

When solving the tasks of processing dynamic distributed workflows, there are problems with access to resources, blocking, limiting the liveliness. The presence of a large number of interacting dynamic distributed workflows poses the task of formal analysis, control and translation, which can be performed

This research is supported by the grant of the Ministry of Education and Science of the Russian Federation, the project № 2.1615.2017.

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.

by various methods. At present, π-calculus is a promising but still very young and evolving theory. It has many open questions and unresolved problems. Petri nets have the following limitations:

• there is no universal framework for project workflows modeling and analysis on the basis of Petri nets. In order to analyze various properties (liveliness, reachable, safety), workflows are modeled in different types of Petri nets, which is "ad hoc".

• there is no mechanism that would assist the designer in modeling and ensure the successful completion of the task with the necessary properties.

The model checking method was extensive use for workflow analysis in error-free systems development during the conceptual design phase. However, it is intended for experienced scientists and engineers, since it is complex in understanding and operating [5].

Dynamic distributed workflows are also specified by managers who do not have training in formal models and informatics. For formal analysis of dynamic distributed workflows, it is required a detailed representation of the process model in a formal language that is difficult to build and understand to managers.

To solve this problem, the authors of the research work propose their own approach to analysis, control and translation into the relevant representations (models) of dynamic distributed workflows.

III. ORCHESTRATION&CHOREOGRAPHY

Orchestration is a description of an enterprise's internal business process in the form of the flow of interaction between internal and external web services of an enterprise. The point of view on this process is purely internal, e.g., from the executive staff's point of view or on from the business process owner's point of view. BPEL (Business Process Executive Language), XPDL (XML Process Definition Language), UML are modeling languages for describing the orchestration. In order to perform an orchestration, it is required the presence of a central processor called web services. In this case, Web services "do not know" that they participate in a more global business process.

The orchestration is always a control from one participant's standpoint in the process. Choreography allows each participant to describe his/her part of the interaction. When using choreography, the messages' sequences between several participants and sources are tracked.

Choreography is the definition of a condition sequence in which several independent participants exchange messages in order to perform some general business task (e.g., B2B). Accordingly, WS-CDL (Web-services Choreography Description Language) and ebXML (Electronic Business using eXtensible Markup Language) are modeling languages for describing the choreography. Business processes choreography does not require a central coordinator, since each web service "knows" when to perform its operations and with what other web service it interacts with (Figure 1).

The proposed standards of orchestration and choreography must meet several requirements related to the language description of business process workflows and to the process execution infrastructure. These requirements include: asynchronous call, management of exceptional situations, transactional integrity assurance, dynamism, flexibility, adaptability, ability of higher-level service composition of existing processes.

Translation programs are used to convert dynamic distributed workflows. For example, a BPEL description of a business process is translated from BPMN (Figure 2), and then the source code in a specific language (e.g., Java) is translated from the BPEL description of the business process. The authors developed RVTt-grammar for this purpose.

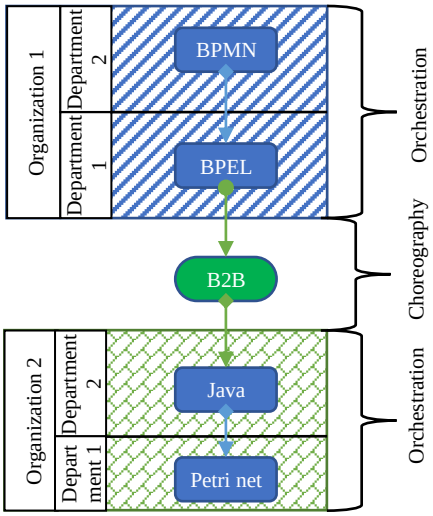

Fig. 1. An example of significance of the orchestration and choreography in large enterprises

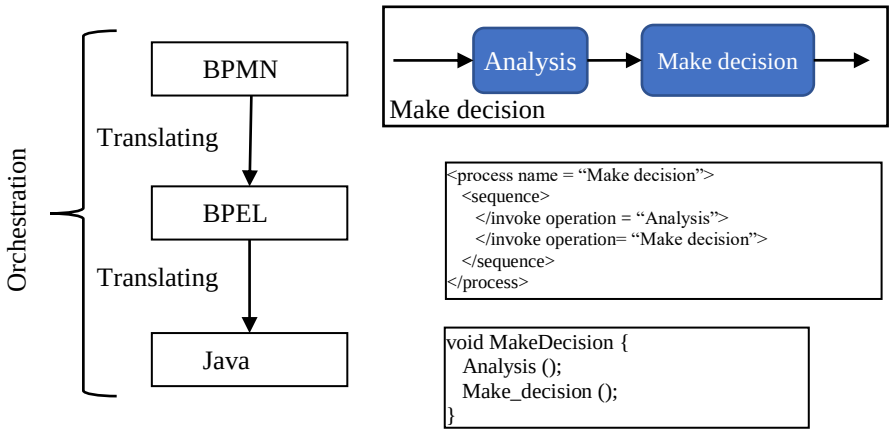

Fig. 2. Translating BPMN to BPEL, and BPEL to Java

IV. RV-, RVTT-GRAMMAR

A. *Review of grammars*

We review a reserved graph grammar, a positional grammar, a relational grammar, and a web grammar to represent business process and workflow as diagrams with the help the tools [6-10].

1. A positional grammar: uses a plex-structure, which has not attaching points, and does not consider dynamically links of inputs\outputs, therefore, it cannot be used for graphical languages by a designer.
2. A relational grammar: generates a non-exhaustive list of errors that cannot be defined to analysis.
3. A reserved graph grammar, and a web grammar: save sentential form of a diagram, that takes a lot of time to analysis it.

For reviewed grammar, we summarize problems as:

1. Increasing rules to describe the grammars takes much time for an analysis, and has exponential or polynomial requires of time.
2. Reviewed grammars have a sentential form to represent a diagram.
3. There are not tools for the semantic analysis of diagram attributes.

B. *RV-grammar*

The authors have developed automaton grammar, called RV-grammar, to analysis and check (control) diagrams for the tools, described in the papers [11-18].

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

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

where $V = \{v_l, l = \overline{1,L}\}$ is an auxiliary alphabet (the alphabetical operations with the internal memory); $\Sigma = \{a_t, t = \overline{1,T}\}$ is a terminal alphabet, which is the union of its graphic objects and links (the set of primitives); $\tilde{\Sigma} = \{\tilde{a}_t, t = \overline{1,\tilde{T}}\}$ 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 RV-grammar (the initial complex of rules), $r_k \in R$ is a final complex of rules.

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

$$\tilde{a}_t \xrightarrow{W\gamma(\gamma_1,\ldots,\gamma_n)} r_m, \quad (2)$$

where $W_\gamma(\gamma_1, \ldots, \gamma_n)$ – n-ary relation, defining an operation with the internal memory depending on $\gamma \in \{0, 1, 2, 3\}$; Ω_μ is a modification operator, changing the type of the operation with the memory, and $\mu\{0,1,2\}$; $r_m \in R$ is the receiver of rules.

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 $\varphi = \alpha_{t1}, \alpha_{t2}, \dots, \alpha_{t\lambda}$ is called RV-output $\alpha_{t\lambda}$ of α_{t1} and it is denoted $\alpha_{t1} \overset{RV}{\Rightarrow} \alpha_{t\lambda}$ if for any $\xi < \lambda$ and $r_e \in R$ are as $\alpha_{t\xi+1} \in r_e \left(a_t \xrightarrow{\Omega_\mu[\lambda_1,\dots,\lambda_n]} r_e \right) \in r_i$.

RV-output is considered to be complete (it is denoted as $\alpha_{t1} \overset{RV}{\Rightarrow} \alpha_{t\lambda}$), if $a_{t\lambda}$ is generated by rules with r_k on the right-hand side.

RV-grammar is effective both for generating and recognizing.

The application of any complex of rules r_0(RV-grammar axiom) generates some chain of language L on its RV-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.

The RV-grammar for the business processes described in notations of BPMN is given in Table 1.

TABLE 1. RV-GRAMMAR FOR BPMN

№	Start State	Quasiterm	End State	Operations with memory
1	r_0	A0	r_1	$W_1(t^{1m^{(k-1)}})$
2		Aim	r_1	O
3		A	r_4	$W_1(t^{2m^{(k-1)}})$
4		EG	r_3	$W_1(t^{3m^{(k-1)}})$
5		EBG	r_5	$W_1(t^{5m^{(k-1)}})$
6		PG	r_3	$W_1(t^{6m^{(k-1)}})$
7	r_1	rel	r_7	O
8	r_2	rel	r_7	O
9		O	r_3	O
10	r_3	assoc	r_9	O
11		O	r_6	O
12	r_4	Aim	r_2	O
13		O	r_3	O
14	r_5	erel	r_8	O
15		O	r_3	O
16	r_6	labelA0	r_7	$W_2(b^{1m})$
17		labelA	r_7	$W_2(b^{2m})$
18		labelEG	r_7	$W_2(b^{3m})$
19		labelIG	r_7	$W_2(b^{4m})$
20		labelEBG	r_5	$W_2(b^{5m})$
21		labelPG	r_7	$W_2(b^{6m})$
22		no_label	r_k	*
23	r_7	Ai	r_1	O
24		Aim	r_2	$W_1(1^{t(1)}, k^{t(2)})/W_2(e^{t(1)})$
25		_Aim	r_2	$W_1(inc(m^{t(1)})/W_3(m^{t(1)} < k^{t(2)})$
26		Ak	r_3	$W_1(1^{t(3)}, k^{t(4)})/W_2(e^{t(4)})$
27		_Ak	r_3	$W_1(inc(m^{t(3)})/W_3(m^{t(3)} < k^{t(4)})$
28		A	r_4	$W_1(1^{t(5)}, k^{t(6)}, (W_2(b^{t(15)}) - 1 + k)^{t(15)}, (W_2(b^{t(16)}) - n + 1)^{t(16)})/W_2(e^{t(5)})$
29		_A	r_4	$W_1(inc(m^{t(5)})/W_3(m^{t(5)} < k^{t(6)})$ && $W_1(t^{2m^{(k)}})/W_3(m^{t(5)}=k^{t(6)})$
30		EG	r_3	$W_1(1^{t(7)}, k^{t(8)}, (W_2(b^{t(16)}) - n + k)^{t(16)})/W_2(e^{t(7)})$
31		_EG	r_3	$W_1(inc(m^{t(7)})/W_3(m^{t(7)} < k^{t(8)})$ && $W_1(t^{3m^{(k)}})/W_3(m^{t(7)}=k^{t(8)})$
32		EBG	r_5	$W_1(1^{t(7)}, k^{t(8)}, (W_2(b^{t(16)}) - n + k)^{t(16)})/W_2(e^{t(7)})$
33		_EBG	r_5	$W_1(inc(m^{t(7)})/W_3(m^{t(7)} < k^{t(8)})$ && $W_1(t^{3m^{(k-1)}})/W_3(m^{t(7)}=k^{t(8)})$
34		PG	r_3	$W_1(1^{t(13)}, k^{t(14)}, (W_2(b^{t(15)}) - n + k)^{t(15)})/W_2(e^{t(13)})$
35		_PG	r_3	$W_1(inc(m^{t(13)})/W_3(m^{t(13)} < k^{t(14)})$ && $W_1(t^{6m^{(k)}})/W_3(m^{t(13)}=k^{t(14)})$
36	r_8	Aim	r_2	O
37	r_9	DO	r_6	O
38	r_k			

C. RVTt-grammar

RVTt-grammar is a translating grammar that allows the system to make a syntax-oriented translation of graphical language diagrams into textual formal descriptions. RVTt-grammar is the development of the RV-grammar, in which the products of the grammar scheme are expanded to store the correspondences in terms of the target formal description, and the internal memory stores the information necessary for the translation.

RVTt - the grammar of the language L (G) is the ordered seven of non-empty sets

$$G = (V, U, \Sigma, \tilde{\Sigma}, M, R, r_0), \quad (3)$$

The following sets are defined additional for (1) in (3).

$M = TT \cup TN$ is a combination of the alphabets of terminal (TT) and nonterminal (TN) symbols of the target language.

$R = \{r_i,\ i = \overline{0, I}\}$ is a grammar scheme of G (the set of names of production complexes, each complex r_i consists of a set of P_{ij} products $r_i = \{P_{ij},\ j = \overline{1, J}\}$);

$r_0 \in R$ is the axiom of RVTt-grammars (the name of the initial set of products), $r_k \in R$ is the final set of products.

The set of V, Σ, $\tilde{\Sigma}$, R, r_0 are inherited from the RV grammar and are used to determine the syntactical correctness of the analyzed diagram.

The sets of U, M are necessary for giving translating functions to the grammar. They create a new formalism by expanding the RV-grammar.

$P_{ij} \in r_i$ production is given as

$$P_{ij}: \bar{a}_t \xrightarrow{\Omega_\mu[W_\nu(\gamma_1,\dots,\gamma_n)]\{\Theta_\mu[W_\nu(\gamma_1,\dots,\gamma_n)]\}} r_m\{\chi\}, \qquad (4)$$

where $W_\nu(\gamma_1, \dots, \gamma_n)$ is n-th ratio that determines the type of operation over the internal memory depending on $\nu \in \{0,1,2,3\}$;

Ω_μ (Θ_μ) is the modification operator in a certain way, changing the type of operation over the memory of the base (target) language, and $\mu \in \{0,1,2\}$;

$r_m \in R$ is the name of the product complex-successor;

χ – is a display of quasi-term in terms of the target language (simbols' set $m \in M$).

V. EFFICIENCY

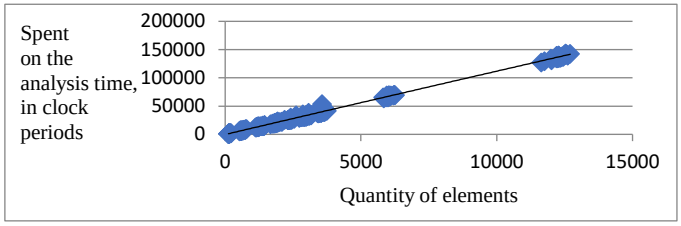

Fig. 3. Efficiency of analysis errors with the help RV-grammar into BPMN

Figure 3 shows the efficiency. Graphic objects are graphic figures as a circle, a rectangle, a rhombus, a square, a line and etc. We propose a formula [13] for calculation of the efficiency that can be written as:

$$\text{Required time} = c \cdot L_s, L_s = \sum_{i=1}^{m}\left(\sum_{j=1}^{V_i} v_{in_{ij}} + \sum_{j=1}^{V_i} v_{out_{ij}}\right) + \sum_{i=m+1}^{t}\left(V_i + \sum_{j=1}^{V_i} v\_out_{ij}\right) + no\_label. \qquad (5)$$

where c – the constant of realization of algorithm, which determines a quantity of time (operators) that are spent to analysis one graphic object; L_s – the number of graphic objects;

V_i – the number of graphic objects of i-type; v_in_{ij} – the number of inputs to j-graphic objects of i-type; v_out_{ij} – the number of outputs from j-graphic objects of i-type; t – a total of object types; m – a quantity of object types that have more than one output.

Thus, RV-grammars have a linear characteristic of time costs, in contrast to known grammars, e.g., positional, preserving, relational, having exponential time analysis.

Regarding error control of RV grammar allow to record the errors called a rupture of a context, connected with use of logical communications, such as 'AND', 'OR', 'XOR', and also context-dependent errors which will not be found by means of the mentioned tools can be in the text (notations) of any diagrams. They become "expensive" errors in design.

CONCLUSION AND FUTURE WORK

In this paper, the authors proposed a framework for the design-engineering workflows analysis and translation. The analysis of existing approaches and tools was carried out and shown the advantage of RV-, RVTt-grammars. In future research works the authors will examine the timed grammars for design-engineering workflows analysis, control and translation in which time factor will occupy a significant place. Using such grammar will eliminate a number of semantic errors at the stage of conceptual design of complex computer-based systems.

REFERENCES

[1] A global Swiss company offering advanced intelligent application software for multiple business sectors. http://whitestein.com/.

[2] 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, August 2014. doi: 10.1007/s10257-014-0256-1

[3] Michael zur Muehlen, Marta Indulska, Gerrit Kamp, "Business Process and Business Rule Modeling Languages for Compliance Management: A Representational Analysis," Proceeding ER '07 Tutorials, posters, panels and industrial contributions at the 26th international conference on Conceptual modeling, vol. 83, pp. 127-132, 2007.

[4] D. Georgakopoulos, M. Hornick, and A. Sheth, "An overview of workflow management: From process modeling to infrastructure for automation," Journal on Distributed and Parallel Database Systems, vol. 3(2), pp. 119-153, April 1995.

[5] Yuan Wang, Yushun Fan, "Using Temporal Logics for Modeling and Analysis of Workflows," Proceedings of E-Commerce Technology for Dynamic E-Business, 2004. IEEE International Conference on, 2004. doi: 10.1109/CEC-EAST.2004.72

[6] Fu K., "Structural methods of pattern recognition," Moscow: Mir, – P.319 1977.

[7] 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.

[8] Wittenburg K., Weitzman L., "Relational grammars: Theory and practice in a visual language interface for process modeling," http://citeseer.ist.psu.edu/wittenburg96relational.html, 1996.

[9] Zhang D. Q., Zhang K., "Reserved graph grammar: A specification tool for diagrammatic VPLs," Visual Languages. Proceedings. 1997 IEEE Symposium on. – IEEE, 1997, pp. 284-291, 1997.

[10] Zhang K. B., Zhang K., Orgun M. A., "Using Graph Grammar to Implement GlobalLayout for A Visual Programming Language Generation System," 2002.

[11] 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) Volume 451 of the series Advances in Intelligent Systems and Computing. Russia, 2016, pp. 395-403.

[12] 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) Volume 450 of the series Advances in Intelligent Systems and Computing. Russia, 2016, pp. 227-236.

[13] 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, 2016, pp. 441-445.

[14] Alexander Afanasyev and Nikolay Voit, “Multi-agent system to analyse manufacturing process models,” Proceedings of International conference on Fuzzy Logic and Intelligent Technologies in Nuclear Science - FLINS2016. France, 2016. pp. 444-449.

[15] Afanasyev A.N., Voit N.N., Voevodin E.Yu., Gainullin R.F., “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, 2015, pp. 285-288.

[16] Sharov O.G., Afanas’ev A.N., “Syntax-directed implementation of visual languages based on automaton graphical grammars,” Programming and Computer Software. 2005. vol. 6. pp. 56-66, 2005.

[17] Sharov O.G., Afanas’ev A. N., “Neutralization of syntax errors in the graphic languages,” Programming and Computer Software. 2008, vol. 1, pp. 61-66, 2008.

[18] Sharov O.G., Afanas’ev A. N., “Methods and tools for translation of graphical diagrams,” Programming and Computer Software. 2011. vol. 3, pp. 65-76, 2011.