ALGORITHMS OF AUTOMATE MODEL CONSTRUCTION FOR BUSINESS GAME EXECUTION SUBSYSTEM

Alexandr Deryabin, Lidia Shestakova, Olga Vikentyeva

Abstract: *Changes of professional environment, caused by introduction of new technologies and techniques, create a necessity in continuous education and development of professional competences. In these conditions, managers and other company employees face the choice of methods and tools of personnel training. A business game is one of the most productive tools of business-education.*

This paper elaborates the ideas embodied previously, which considered the conceptual approach to a toolkit creation for active training techniques in a form of competence-based business game studio. Competence-based business game studio is an ergatic system for development of professional competences, which are required to ensure organization's business processes.

Business game control is performed with the help of automate module, which executes the interpretation of expressions in algorithm logical scheme language.

Unified business process is the input data for automate model construction. Unified business process is acquired from the information about company's real business processes, represented in poorly formalized ways. Unified business process transforms into unified training business process, which includes possible business game trainee actions and information about input, output, administrative data and business process operation execution mechanism. The next step to automate model is the construction of scenario graph, which represents a more formalized description of unified training business process. Next, the administrating algorithm of the business game in algorithm logical scheme language is built. The paper considers the algorithms of transition between models that allow automating the process of acquiring algorithm logical scheme in order to implement the business games scenario.

Keywords: *active learning methods, business-game, business-process, automate model, algorithm logical scheme, algorithm, metamodels*

ACM Classification Keywords: *K.3 Computers and Education: K.3.2 Computer and Information Science Education – Information systems education. K.4 Computers and Society: K.4.3 Organizational Impacts – Employment. I. Computing Methodologies: I.2 Artificial Intelligence: I.2.1 Applications and Expert Systems – Games.*

Introduction

Changes of professional environment, caused by introduction of new technologies and techniques, create a necessity in continuous education and development of professional competences. In these conditions, managers and other company employees face the choice of methods and tools of personnel training. Currently, e-learning systems have gained much popularity. Implementation of such systems enables to cut costs related to training organization in companies and educational institutions through process automation and use of Internet technologies. However, the majority of e-learning systems only support the traditional education model slightly modifying it with contemporary communication technologies. Hence, a certain trend must be noticed, the trend implies use of active training techniques and business games in particular [Zichermann, 2011], [Wells,1990]. A business game (BG) is one of the most productive tools of business-education. Business games allow trainees to acquire hands-on experience, fosters professional competences and teamwork efficiency in real-like conditions.

This paper elaborates the ideas previously embodied in [Deryabin, 2013], [Викентьева, 2013], [Викентьева, 2014], which considered the conceptual approach to a toolkit creation for active training techniques in a form of competence-based business game studio (CBGS). CBGS is an ergatic system for development of professional competences, which are required to ensure organization's business processes.

Domain formalization for automate model construction

Automate model is constituted by a sequence of operators in algorithm logical schema (ALS) language, which implements the algorithm of business game control. Any expression in ALS language may be represented as following:

L = {H, A, P, ω, ↑, ↓, K}, where,

H – algorithm start operator;

K – algorithm finish operator;

P – conditional transfer;

A – control action;

ω– unconditional transfer;

↑ - transfer beginning;

↓ - transfer end.

BG control is performed with the help of automate module, which executes the interpretation of ALS expressions. Furthermore, the module recognizes each consecutive operator and either conveys subsequent game scene code to the operational module, or carries out the conditional or unconditional transfer to the next operator, depending on the game status code received from the operational module.

Unified business process (UBP) is the input data for automate model construction. UBP is acquired from the information about company's real business processes, represented in poorly formalized ways (graphical models, text description, and regulatory documents). UBP transforms into unified training business process (UTBP), which includes map of operations that contains a tree of possible business process operations (BG trainee actions) and operation model, which contains information about input, output, administrative data and business process operation execution mechanism. The map of operations also includes decision making points, which stand for player (user) interaction during the business game. Figure 1 uses following notation: UBP – unified business process; UTBP – unified training business process; RBP– real business process; MOp – map of operations; OP – operation model; DMP – decision making point; SG – scenario graph; ALS – algorithm logical schema; AM – automate model; OM – operational model.

After the construction of UTBP the process of BG design may be separated into two steps:

- operational model construction, which includes scene model, resource model and screen model;
- automate model construction, which is needed to build ALS expressions control the business game.

This paper considers the process of automate model construction and algorithms used for UTBP's map of operations transformation into a sequence of models. This sequence is required to build ALS expressions.

DSM-platform Metaedit+ [Mazanek, 2008], [Сухов, 2009] was used to develop interrelated metamodels for UTBP description: "Operation", "Map of Operations" and "Decision Making Point".

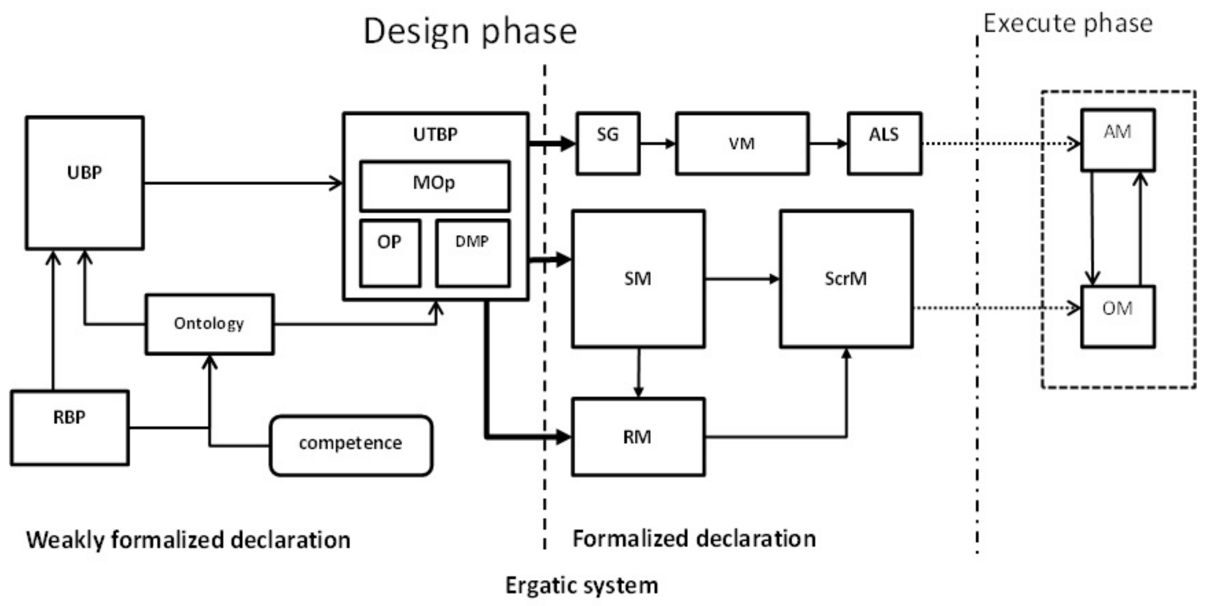


Figure 1. Domain formalization for automate model construction

“Operation” metamodel describes separate operations (tasks), which constitute the business process and include resources (informational, financial, manpower), equipment, actors etc.

“Map of Operations” metamodel enables to describe business process as a multivariant sequence of operations and moments where player has to make decisions.

“Decision Making Point” metamodel allows to describe the process of decision-making, unfolding it through a sequence of reactions.

Let us consider an instance of streamlined business process (Figure 2), which consists of actions D1, D2, D3, D4 and business condition BC1 that determines the rule of business process execution.

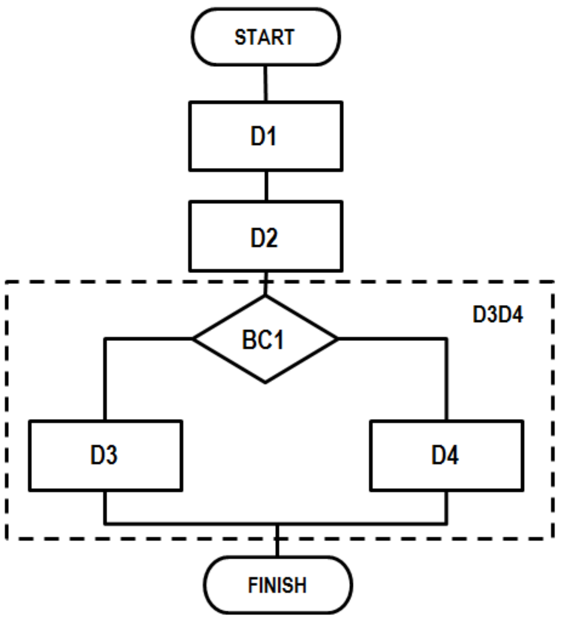


Figure 2. The process of transformations from UBP to unified training business process

Let us build the map of operations for the business process (Figure 2) with the help of “Map of Operations” and “Decision Making Point” metamodels. The map of operations represents a tree, which includes decision making points S and actions D. Decision making point enables transfer from one action to another, with transfer from a single point to multiple actions being possible. Each branch of the tree may contain action Di only once. Thus, the result is a hierarchical structure, where each branch represents a possible business game path.

UBP may be quite difficult and include not only sequential operations, but various business conditions and recurring operations as well. To build the UTBP, it is proposed to replace the conditional and cyclical structures with generalized blocks. This enables to describe the UBTP as a linear sequence of operations. A new map of operations is the built for each generalized block.

Hence, the algorithm of UBP to map of operations transformation may be as following:

1. All blocks with business conditions in the UBP must be replaced with generalized blocks. This also applies to linear block sequences if needed. The replacement must be carried out in such way that the SBP may be executed linearly (convert the UBP to a linear form). In result, the UBP will contain only generalized actions D1, …, Dn.

2. Build start and finish blocks.

3. Build a list of possible actions D = {D1, …, Dn, Dk}, where Dk – transfer to the finish block, Di – execution of action i.

4. Build decision making point (DMP) S0.

5. Build branches for the current DMP regarding the actions from list D.

6. For each branch built, except the branches with transfer to the finish block:

    6.1. Build a list of possible actions, that does not contain Di: D={D1,…, Dn, Dk}\{Di}.

    6.2. If list D={Dk}, build a transfer to the finish block and return to step 6, otherwise go to step 6.3.

    6.3. Build decision making point St.

    6.4. Build branches regarding the actions from list D.

    6.5. Return to step 6.

Figure 3 depicts the result of algorithm application to the UBP (Figure 2). Generalized blocks with business conditions have the same structure. Map of operations of a block must contain a DMP (Si0), where the player would be able to choose different courses of actions with or without condition testing. If the test is present, it is possible to pick one of the two actions: test is positive (BC=1) or test in negative (BC=0). In both cases DMPs are used to choose actions. Execution of a generalized block ends with a DMP to alleviate the introduction of the block into the general map of operations. More complicated cases, which imply execution of several actions in one block, are resolved by contraction of operations into a single generalized block. Figure 4 shows the map of operations for generalized block D3D4 (Figure 3) with business conditions (Figure 2).

The next step to automate model is the construction of scenario graph (SG), which represents a more formalized description of UTBP map of operations.

Scenario graph (Figure 5) is a graph, with BG scenes as graph’s vertexes. These scenes contain conditions of transitions and corresponding DMPs of the UTBP. Edges of the graph comply with the actions, taken during the transitions between scenes. The amount of edges and vertexes is large enough to employ graph representation that includes a shared bus with indexed edges.

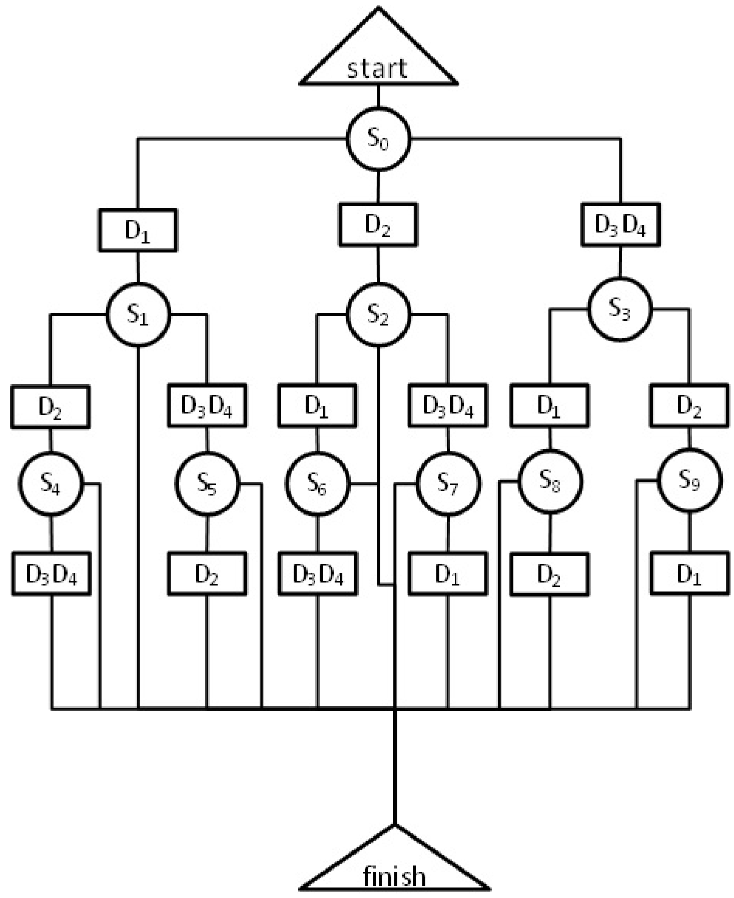


Figure 3. UTBP Map of Operations

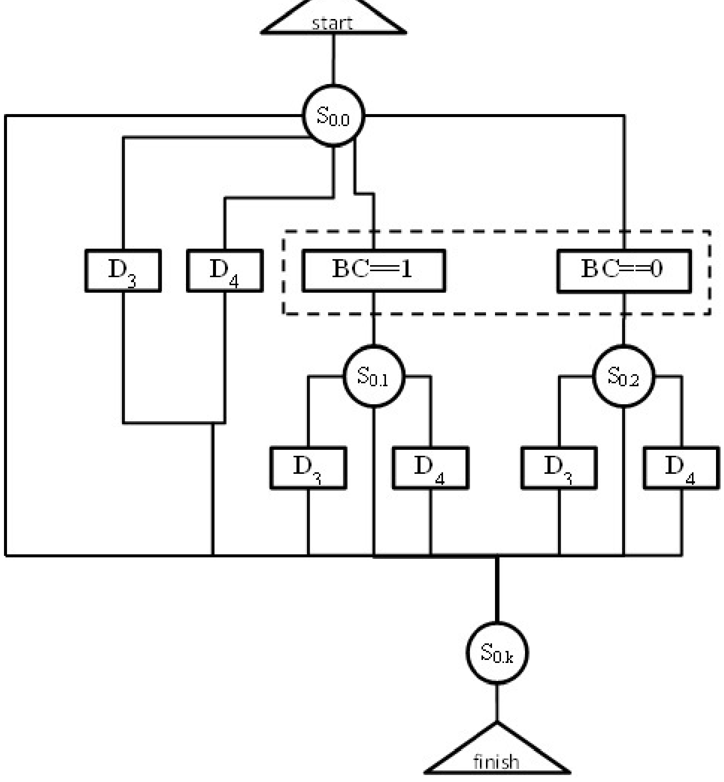


Figure 4. Generalized Block's Map of Operations

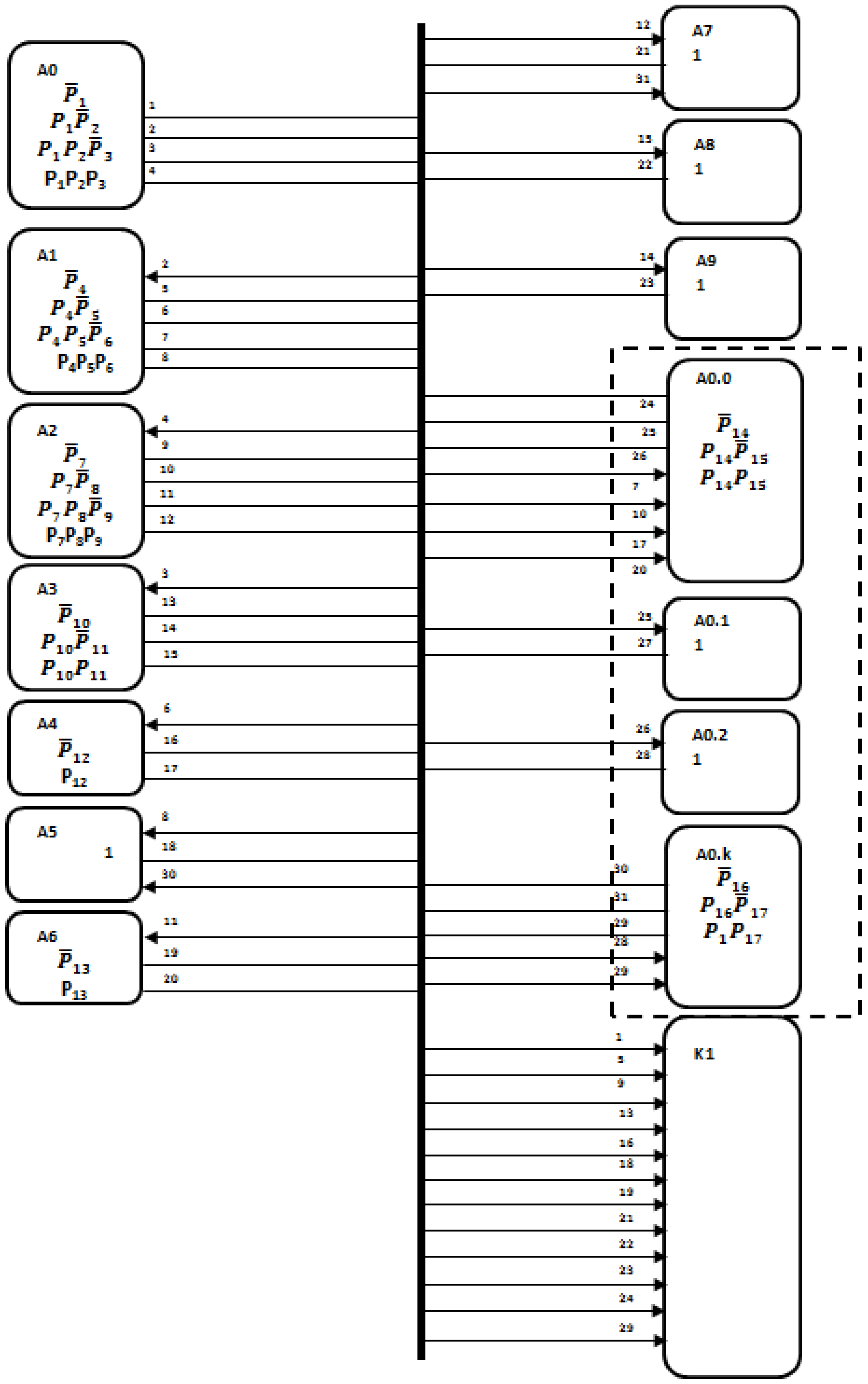


Figure 5. Scenario Graph

Let us consider the algorithm of SG construction:

1. Each decision making point (DMP) of the UTBP map of operations is put at the corresponding SG vertex Ai.

2. For every action from the action list D of each DMP in the map of operations a set of conditions P must be made. The set is written down to the corresponding SG vertex.

2.1. The amount of elements in condition set is defined as $\mathrm{k_P} = \log_2(\mathrm{k_D}) + 1$, where $\mathrm{k_D}$ – amount of elements in list D.

2.2. The set of conditions is formed as following:

2.2.1. Define index of the first condition in the set *as [index of the last condition in previous DMP + 1* and generate indexes for each condition in the set {*P1, P2, …, Pk$_p$+1*}.

2.2.2. Define condition values in the set: first condition – $\overline{\mathrm{P1}}$, second condition – $\mathrm{P1}\overline{\mathrm{P2}}$, third condition $\mathrm{P1P2}\overline{\mathrm{P3}}$, last condition – $\mathrm{P1P2}, \dots \mathrm{Pk_P} + 1$. $\overline{\mathrm{Pi}}$ – negation of condition $\overline{\mathrm{Pi}}$.

3. For each SG vertex build outgoing edges corresponding to the conditions of the previously formed set. Then assign each edge sequential indexes starting with 1.

4. Each outgoing edge has a corresponding entering edge with the same index, which is built according to the following rule: each SG vertex Ai has a corresponding DMP, represent the connection between DMPs Si and Sj as a connection between SG vertexes Ai and Aj and build an entering edge. Entering edges have direction.

5. If a branch connection DMPs has a generalized block, do as following:

5.1. For the generalized block a SG vertex Bk is built (0<=k<=n, n – amount of generalized blocks). Entering edges correspond to edges outgoing from vertex Ai. Outgoing edge correspond to edges entering vertex Aj.

5.2. Generalized block is represented as a SG with vertexes Aij, where i – index of the block in a sequence of generalized blocks, j – index of a DMP in the generalized block i.

5.3. Edges entering Bk transform into edges entering block Ai0.

5.4. Edges outgoing from Bk transform into edges outgoing from block Aik.

5.5. Дуги, входящие в Bk, преобразуются во входные дуги блока Ai0.

Thus, SG contains conditions for transitions between BG scenes. To be able to transform a SG into an algorithm in ALS language that can executed by the automate module, it is needed to build a vertex-adjacency matrix (Figure 6). Matrix elements are represented by the sets of conditions, which are tested during transitions between SG vertexes.

Vertex-adjacency matrix is built as following:

1. Fill the matrix with zeroes.

2. If the transition between vertexes Ai and Aj exists, then the corresponding matrix cell is filled with the transition condition P

3. If the transition between vertexes Ai and Aj exists and is unconditional, the corresponding matrix cell is filled with number 1.

	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	K1	V0	V1	V2	Vk
A0	0	$P_1\overline{P}_2$	$P_1P_2\overline{P}_3$	$P_1P_2P_3$	0	0	0	0	0	0	$\overline{P}_1$	0	0	0	0
A1	0	0	0	0	$P_4\overline{P}_5$	$P_4P_5\overline{P}_6$	0	0	0	0	$\acute{P}_4$	$P_4P_5P_6$	0	0	0
A2	0	0	0	0	0	0	$P_7P_8\overline{P}_9$	$P_7P_8P_9$	0	0	$\overline{P}_7$	$P_7\overline{P}_8$	0	0	0
A3	0	0	0	0	0	0	0	0	$P_{10}P_{11}$	$P_{10}\overline{P}_{11}$	$\overline{P}_{10}$	0	0	0	0
A4	0	0	0	0	0	0	0	0	0	0	$\overline{P}_{12}$	P_{12}	0	0	0
A5	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
A6	0	0	0	0	0	0	0	0	0	0	$\overline{P}_{13}$	P_{13}	0	0	0
A7	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
A8	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
A9	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
K1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V0	0	0	0	0	0	0	0	0	0	0	$\overline{P}_{14}$	0	$P_{14}P_{15}$	$P_{14}\overline{P}_{15}$	0
V1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
V2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Vk	0	0	0	0	0	$P_{16}P_{17}$	0	$P_{16}\overline{P}_{17}$	0	0	$\overline{P}_{16}$	0	0	0	0

Figure 6. Vertex-adjacency matrix with logical

The base of conditions for vertex Ai is the condition set P with the maximum length in the vertex-adjacency matrix row. The set P must correspond to vertex Ai. For instance, for vertex A0 (fig. 6) base of conditions is represented by set P1P2P3.

Using the vertex-adjacency matrix it possible to derive the administrating algorithm of the BG in ALS language as following:

1. Write the start and finish operators (H, K).

2. Write SG vertex names (Ai) between operators H and K.

3. For each operator Ai write base of conditions and operator of unconditional transition ω (coincides with 1 in the vertex-adjacency matrix).

4. After each condition and operator of unconditional transition write ↑. For instance, base of conditions and transition operators for operator A0 will be P1↑P2↑P3↑ω↑.

5. In front of every operator Ai put operator ↓.

Indexes of operators ↑ and ↓ (transitions between Ai and Aj) are defined as following:

6. Initial index t=0.

7. From A0 to K:

  7.1. Consider every condition Pk in string Aj and assign index t for Pk↑t.

  7.2. If the Ai and Aj cells of adjacency matrix have $\overline{\mathrm{Pk}}$, then the transition between Ai and Aj is executed according to this condition. In this case it is necessary to assign index t to operator ↓, which stands in front of operator Aj, where Aj is the column of the matrix cell with condition $\overline{\mathrm{Pk}}$. Then increase counter t by 1.

  If the Ai, Aj cell has no negated conditions, then the transition from Ai to Aj is executed through operator of unconditional transition with the corresponding index. Increase counter t by 1.

  If the transition is executed towards already indexed operator ↓, then this index does not change, rather the index of operator ↑ is changed and counter t is not increased.

Figure 7 shows an algorithm logical schema built with the help of adjacency matrix (Figure 6). The derived algorithm is then loaded into a software module, which is then used to administrate the business game.

$$H_1\ A_0\ P_1\uparrow^1\ P_2\uparrow^2\ P_3\uparrow^3\ \omega\uparrow^4\ \downarrow^2\ A_1\ P_4\uparrow^1\ P_5\uparrow^5\ P_6\uparrow^6\ \omega\uparrow^7\ \downarrow^3\ A_2\ P_7\uparrow^1\ P_8\uparrow^7\ P_9\uparrow^8\ \omega\uparrow^9\ \downarrow^4\ A_3\ P_{10}\uparrow^1\ P_{11}\uparrow^{10}\ \omega\uparrow^{11}\ \downarrow^5\ A_4\ P^{12}\uparrow^1\ \omega\uparrow^7\ \downarrow^6\ A_5\ \omega\uparrow^1\downarrow^8\ A_6\ P_{13}\uparrow^1\ \omega\uparrow^7\ \downarrow^9\ A_7\ \omega\uparrow^1\ \downarrow^{11}\ A_8\ \omega\uparrow^1\ \downarrow^{10}\ A_9\ \omega\uparrow^1\ \downarrow^7\ A_{0.0}\ P_{14}\uparrow^1\ P_{15}\uparrow^{12}\ \omega\uparrow^{13}\ \downarrow^{13}\ A_{0.1}\ \omega\uparrow^{14}\ \downarrow^{12}\ A_{0.2}\ \omega\uparrow^{14}\ \downarrow^{14}\ A_{0.k}\ P_{16}\uparrow^1\ P_{17}\uparrow^9\ \omega\uparrow^6\ \downarrow^1\ K_1$$

Figure 7. Algorithm logical scheme

Thus, the step-by-step transition from UBP to business game automate model has been considered.

Conclusion

One of the problems faced by business game developers is the problem of complicated domain formalization. It is needed to derive a range of models to get from a real business process of an organization to a set of business game resources. Business game control is performed with the help of automate module, which uses automate model of the business game in form of algorithm logical schema as input data. The automate model is acquired as a result of gradual transition from unified business process model to unified training business process, which serves as a basis for business game scenario graph. The construction of automate model is finished by building the algorithm logical schema derived from analysis of vertex-adjacency matrix that corresponds to the scenario graph. This paper proposes construction algorithms of unified business process, scenario graph and algorithm logical scheme.

Bibliography

[Deryabin, 2013] Deryabin A.I., Shestakova L.V., Vikentyeva O.L.. The Construction of competency-based business game operational model // International Journal "Information Technologies & Knowledge". 2013. Vol. 7. No. 4. P. 303-313.

[Mazanek, 2008] Mazanek S. Visual Languages. MetaEdit+ : [Электронный документ] (http://visual-languages.blogspot.com/2007/11/metaedit.html).

[Wells, 1990] R. A. Wells. Management Games and Simulations in Management Development: An Introduction // Journal of Management Development. 1990. Vol. 9, Issue 2. P.4-6.

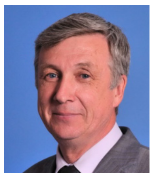
[Zichermann, 2011] Zichermann G. and Cunningham C. Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps [Book]. - Sebastopol, California : O'Reilly Media, 2011.

[Викентьева, 2014] Викентьева, О.Л. Формализация предметной области при проектировании деловой игры] / О.Л. Викентьева, А.И. Дерябин, Л.В. Шестакова // Информатизация и связь. – № 1, 2014. – С. 58-61.

[Викентьева, 2013] Викентьева, О.Л. Концепция студии компетентностных деловых игр [Электронный ресур] / О.Л. Викентьева, А.И. Дерябин, Л.В. Шестакова // Современные проблемы науки и образования. – 2013. – № 2; URL: http://www.science-education.ru/108-8746 (дата обращения: 03.04.2013).

[Сухов, 2009] Сухов А.О. Среда разработки визуальных предметно-ориентированных языков моделирования // Математика программных систем: Межвуз. сб. научн. тр. / Перм. ун-т. Пермь, 2008. С. 84-94.

Authors' Information



Alexander Deryabin *– National Research University Higher School of Economics, City of Perm, Perm, Russia, e-mail: paid2@yandex.ru*

Major Fields of Scientific Research: General theoretical information research, Multi-dimesional information systems

Lidia Shestakova *– National Research University Higher School of Economics, City of Perm, Perm, Russia, e-mail: L.V.Shestakova@gmail.com*

Major Fields of Scientific Research: General theoretical information research, Multi-dimensional information systems

Olga Vikentyeva *– National Research University Higher School of Economics, City of Perm, Perm, Russia, e-mail: oleovic@rambler.ru*

Major Fields of Scientific Research: General theoretical information research, Multi-dimensional information systems